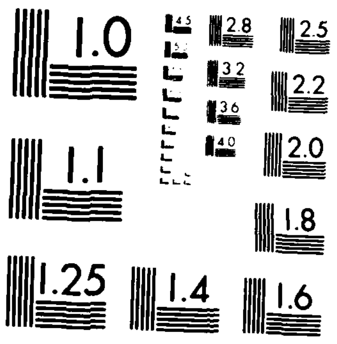
AD-A152 116   A HEURISTIC SOLUTION TO THE SPACING PROBLEM: MAXIMIZING   1/1
THE MINIMUM EUCLI..(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI..   R E ROLEY
UNCLASSIFIED   DEC 84 AFIT/GOR/OS/84D-10   F/G 9/2   NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

①

AFIT/GOR/OS/84D-10

A HEURISTIC SOLUTION TO THE SPACING PROBLEM: MAXIMIZING THE MINIMUM EUCLIDEAN DISTANCE AMONG N POINTS IN A CONVEX REGION

THESIS

Ross E. Roley
Captain, USAF

AFIT/GOR/OS/84D-10

Approved for public release; distribution unlimited

AFIT/GOR/OS/84D-10

A HEURISTIC SOLUTION TO THE SPACING PROBLEM: MAXIMIZING THE MINIMUM EUCLIDEAN DISTANCE AMONG N POINTS IN A CONVEX REGION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Ross E. Roley, B.S.

Captain, USAF

December 1984

Preface

This section is normally reserved for words of gratitude to wife, God, advisor, parents, or friends for their respective inputs to one's life and thesis. Because I have no wife or friends, and am not religious, I will be limited to talking about my parents and advisor. Thanks, Mom and Dad, for teaching me the proper morals of honesty, hard work, and the value of a dollar. Because of that, you saved yourselves a lot of money when I chose to attend the Air Force Academy which is honorable, tough, and free. Actually, I never say often enough how much I love you, so if you ever doubt it, read this Preface. I love you. Thanks also goes to my advisor, Ivy Cook, because of his tremendous patience and assistance with my thesis. Thanks also to David Post of the Air Force Aerospace Medical Research Laboratory (AFAMRL) for his assistance with my thesis. "Just doing my job," he says. Not true, I say.

Ross E. Roley

Table of Contents

List of Figures

List of Tables

Abstract

This [illegible] ~~report~~ introduces a new method for solving the problem of optimally spacing points in a three-dimensional region so that their distances from each other are as great as possible. One application of the problem deals with color selection for aircraft displays where the colors are plotted as points in a three-dimensional color space and the distance between two points is directly related to the distinguishability of the two colors. The method itself is a heuristic algorithm very similar to one designed by Carter and Carter (2). The newer algorithm apparently yields similar solutions with fewer runs, but because it is more thorough, it is slower. The program was tested on problems as large as 23 points whose feasible region had seven faces. The major disadvantage of this new method is that its solutions are not guaranteed to be optimal. As a result, the user must perform several replications of various randomly selected starting locations in order to increase the chances of achieving an optimal solution.

I. Introduction

For centuries, mathematicians have tried to solve tricky little puzzles that perplex and confuse the human mind. Rubick's Cube is a popular recent example of this phenomenon. This thesis is another example. Simply put, the problem addressed herein is to space points in a three-dimensional region (like a box) so that they are as far apart from each other as possible. On the face of it, the task seems relatively easy, but nothing could be further from the truth. A simple example would be to maximize the distance between just two points in a cube. While the answer to that problem might be intuitively obvious (place the points in opposite corners), add one more point and the solution is not so apparent and requires a great deal of mathematical rigor.

Faced with a problem like this in real life, most of us would be content to settle on something that is less than optimal. But, to a pilot who needs to distinguish between friendly and hostile forces on his aircraft display, the importance of an optimal solution takes on a different meaning. That's why this project is so useful, and I sincerely hope that it will help that pilot in even a small way.

One final thought before moving on. Although this is a Master's thesis on what is potentially a very technical subject, it is my desire to make the report understandable to even the most uninformed of laymen, and perhaps even enjoyable at times. I do not believe that writings at this level were meant to be boring and esoteric. So, if you cannot understand it, then I have failed and I should be shot at dawn. But if you don't _try_ to understand it, then you're already wearing the blindfold and should be put in front of the firing squad instead. Load the rifle, cock the trigger, and turn the page to find out who's shooting who.

II. Background

Why in the world would one want to space points in a three-dimensional region, anyway? Well, there are many reasons. Say you wanted to locate speakers for a quadraphonic stereo system in your living room. You would want them as far apart from each other as possible because you don't want the various sounds to become confounded. Or perhaps you own a warehouse with a limited number of security cameras. You would probably want the cameras optimally spaced so that no two were filming the same areas of the warehouse. How would you go about solving this problem mathematically? You could try to maximize the sum of the distances between the objects, but that might result in a solution with an unacceptably small distance. A more desirable solution would probably result from maximizing the distance between the two closest objects. This "maximin" formulation is accepted as the objective for the spacing problem.

The spacing problem is very similar to a problem known in the world of operations research as the obnoxious facility location problem. An example of an obnoxious facility would be a nuclear waste disposal site which is most desirable when it is located far away from cities. Church and Garfinkel (5) offer a solution to the obnoxious facility location problem where the facilities can be placed in a discrete number of locations on a network, but the differences between that problem and the spacing problem preclude successfully adapting Church and Garfinkel's method to the spacing problem. First of all, the idea is to locate obnoxious facilities as far away as possible from other <u>fixed points</u>, whereas the spacing problem aims to locate the objects as far away from each other as possible. Second, Church and Garfinkel's facilities can be placed in only a finite number of locations on a network, while the objects in the spacing problem can take on an infinite number of locations. Finally, the spacing problem's objective is to maximize the minimum distance between points (maximin), as opposed to Church and Garfinkel's objective of maximizing the median distance (maxian).

Another problem similar to the spacing problem is the location problem. It seeks to locate objects (like warehouses) as close to other fixed facilities (such as retail stores) as possible. Once again, much work has been done in the area by many people, most notably Charalambous (1 and 3), Cooper (6), Juel (8 and 11), and Love (10, 11, and 12), but any applicability to the spacing problem is negligible. As before, there is the problem of locating facilities in relation to fixed objects instead of each other. The location problem is also concerned with minimizing the sum of the distances (minisum) or minimizing the maximum distance (minimax), not with maximizing the minimum distance.

There are many practical applications of the spacing problem. In addition to those mentioned before, the spacing problem could be applied toward locating MX missile silos, spacing mines in a mine field, or perhaps placing communications satellites in space for maximum coverage of the earth. A particularly fascinating application of the spacing problem is in the area of color spacing. One of the tasks of the Air Force Aerospace Medical Research Laboratory (AFAMRL) has been to choose colors for aircraft displays, air traffic control displays, aeronautical maps, etc., so that all the colors are as distinguishable from each other as possible. When one is dealing with a color Cathode-Ray Tube (CRT) display, the CRT's red, green, and blue guns are used additively to produce various colors. Therefore, any color can be uniquely defined by three parameters. They are the luminances (brightnesses) of the CRT's red, green, and blue guns. So, each color produced by the CRT can be plotted as a point in three dimensions where the axes represent those three values. A depiction of the resulting color space is shown in Figure 1. The boundaries of the region correspond to the technological limits of the CRT. For instance, in Figure 1, the maximum luminances are 50, 160, and 20 candellas per meter squarred for the red, green, and blue guns, respectively, with zero being the minimum. In addition, there is a small region near the origin that is not a feasible color choice for this example because dark colors are not desired.

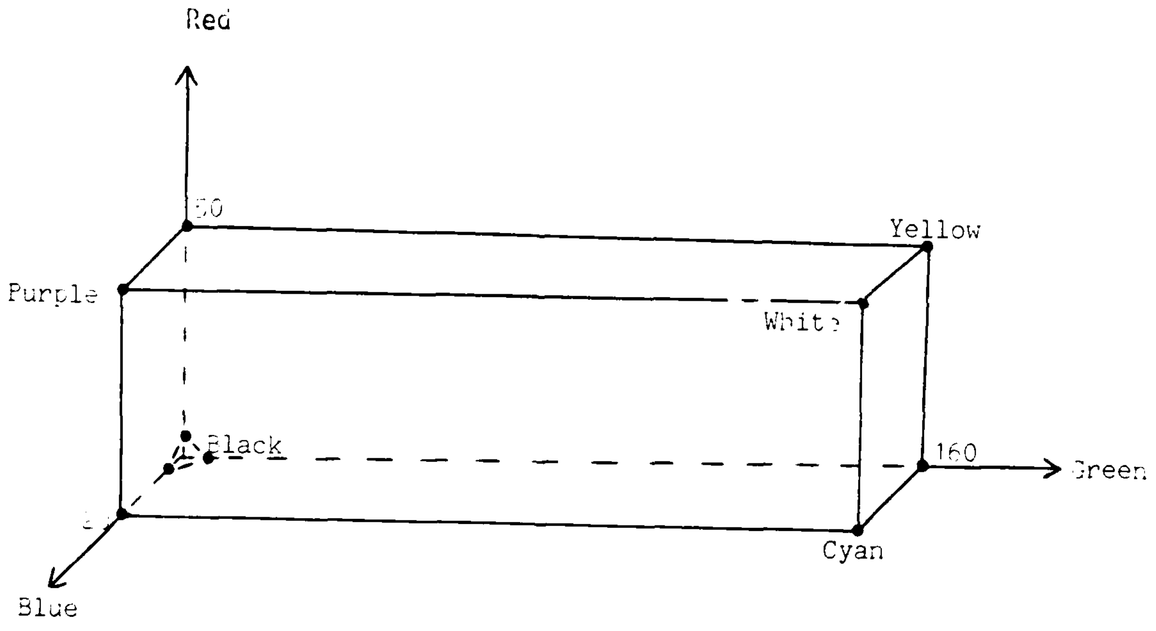


Figure 1. Example Feasible Region for the Color Problem

Unfortunately, this red, green, blue color space is not "perceptually uniform." That is to say that the perceived difference between the red and the purple shown in Figure 1 is not the same as the difference between yellow and white even though their distances are the same. The Commission Internationale de l'Esirage (CIE) recommends the CIE L*u*v* system as a more perceptually uniform color space (13). The L* axis is a function of the luminance of a color, while the u*v* plane identifies the color's position in a transformation of the standard u´v´ CIE color diagram (see Figure 2). The distinguishability of two

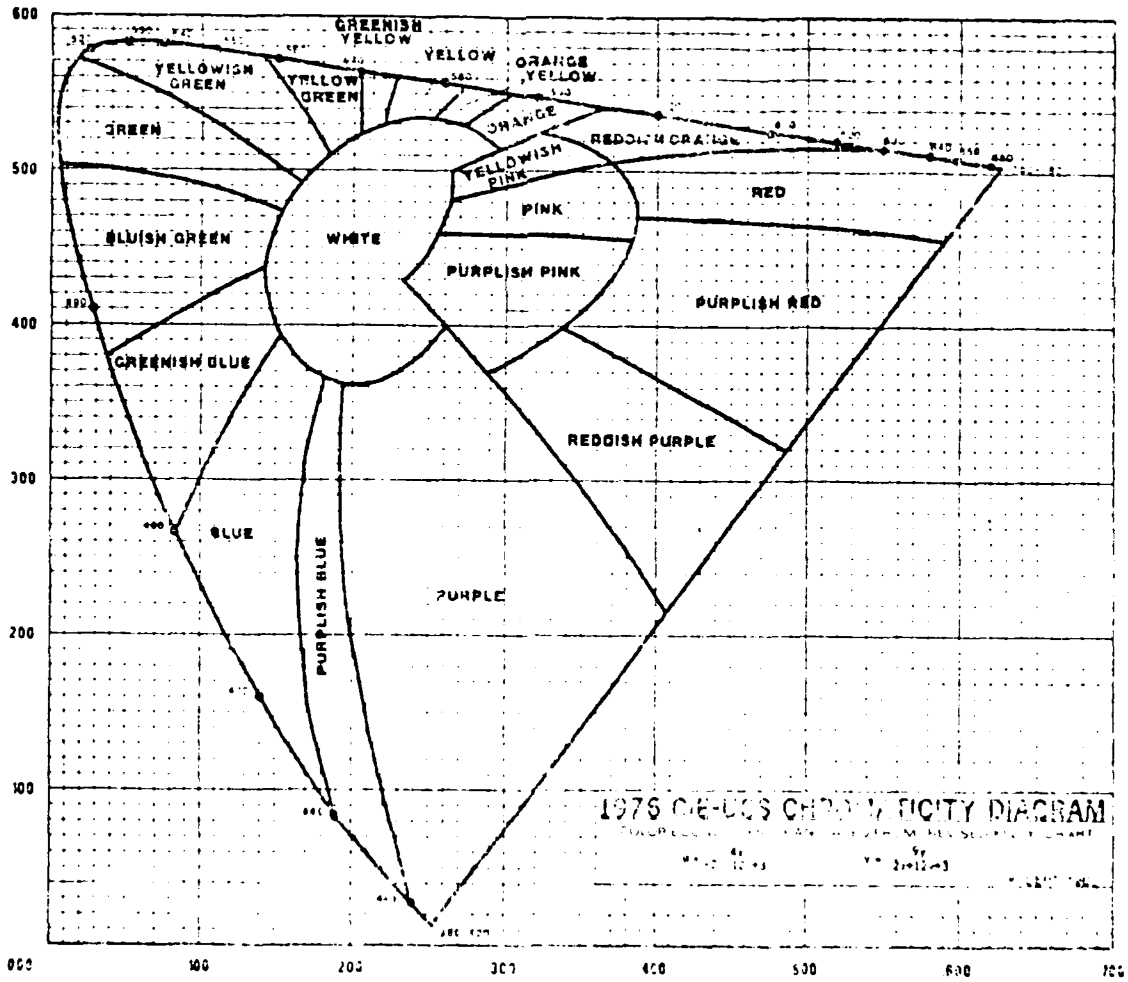


Figure 2. Color Chart for 1976 CIE-UCS Chromaticity Diagram

colors in the CIE L*u*v* space is approximately proportional to the Euclidean distance between their points. Euclidean distance is nothing more than our common sense notion of distance whose formula is given by:

$$d(i,j) = \sqrt{(L_i^* - L_j^*)^2 + (u_i^* - u_j^*)^2 + (v_i^* - v_j^*)^2} \qquad (1)$$

where

$d(i,j)$ = the distance from point i to point j

(L_i^*, u_i^*, v_i^*) = the coordinates of point i

Carter and Carter have developed a computer algorithm to solve the color spacing problem and cite several military applications (2:2936). They describe how the method could be used to choose colors for strategic aircraft displays where the different colors correspond to friendly, hostile, and neutral forces or various enemy target types. As another example, an air traffic controller's display can show airplanes at different altitudes all represented by various colors. Some engineering applications include showing "the distribution of some property throughout a system, such as stress in a structure or percent of full capacity in various parts of an electric power grid" (2:2936). As you can see, there are many interesting applications of the color spacing problem both in the military and in the private sector. AFAMRL, however, has not been able to successfully implement Carter and Carter's program because of various bugs and program deficiencies and has given only educated answers when decisions regarding color selection were required in the past. Concerning the algorithm, Carter and Carter themselves believe that "presumably a more efficient one could be devised by specialists in operations research" (2:2937). For those reasons, this effort was undertaken. The general objectives herein are to provide AFAMRL with a working program that produces good answers to the color spacing problem, and is also an improvement over Carter and Carter's method.

III. <u>Literature Review of Carter and Carter's Method</u>

The only known solution to the color-spacing problem was introduced by Carter and Carter in 1982 (2). It consists of first randomly placing n points in the region (shown in Figure 1), then identifying the minimum CIE L*u*v* distance between all n(n-1)/2 pairs of points in that region. Let's call that value D. Once that distance has been identified, the two closest points (i.e., the endpoints associated with the minimum distance) are investigated to see what effect is created by moving each endpoint to its 26 adjacent locations. This step can conceptually be thought of as "wiggling" the two closest points to see if they can be moved farther apart. Figure 3 shows the various alternatives for one endpoint. The alternatives fall on the boundaries of a cube whose sides are twice the step-size in length and whose center (the 27th point) is the endpoint itself. There are 52 different moves that can be

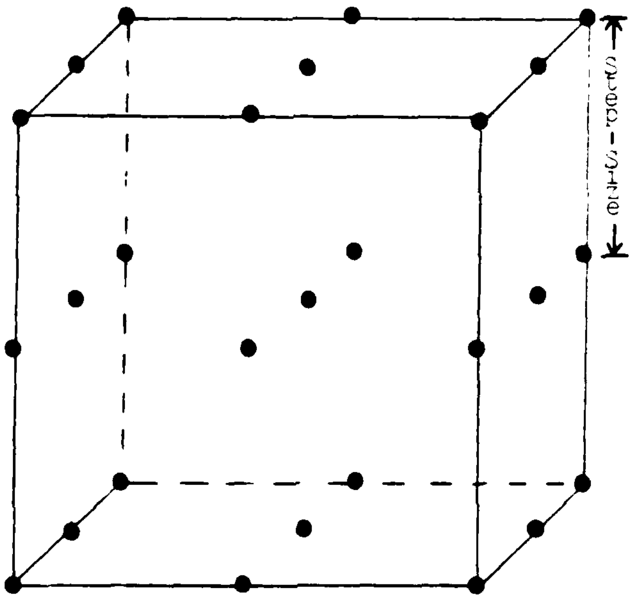


Figure 3. Alternative Locations for an Endpoint

made (26 for each endpoint). If a move decreases the distance between those two points or pushes one of them outside of the region, that alternative is no longer considered. The remaining alternatives are then ranked according to how much they increase the distance between the endpoints.

If the highest ranking alternative causes an increase in D, that move is made. Otherwise, the second highest ranking alternative is considered, and so on, until one of the alternatives causes an increase in D. If none of the alternative moves increase D, then the step-size is halved and the process is repeated; otherwise, the point is moved and the process is repeated until an expanding move cannot be made, at which time the step-size is halved.

This halving continues until the step-size is less than one luminance unit in the red, green, and blue color space. Once the step-size is less than one unit, it is increased back to its original value and the point-moving and step-size halving process is repeated, "to check that the value of D arrived at is not merely a local optimum" (2:2938). If the points remain in the same place throughout the repetition, then "the ... configuration of points is assumed to represent a global optimum value of D" (2:2938).

In their results, the authors state that when the number of colors (n) is three, the solution is identical for each random placement of points, and that, in general, the number of identical solutions decreases as n increases. They also give the results of example problems where 3, 4, 6, 10, and 25 colors were spaced.

Carter and Carter's method is a very good first attempt. Nonetheless, it h[illegible] some problems as well. For instance, there is no supporting proof that the solutions are globally optimal, although Carter and Carter assume that they are (2:2938). In the article, the authors summarize results of their procedure where many replications are made for placement of 3, 4, 6, 10 and 25 points. For the case of six points, 50 replications were made and the variance among the values of D arrived at was 62.45, where the maximum value of D for all replications was 124.08 (2:2938). This relatively high variance points to the fact that a wide range of values for D can be expected for any one replication, and that most replications are not close to even the maximum known value of D for that problem, let alone the global optimum value which for color spacing problems is largely still unknown. Whether or not the solutions represent even local optima is uncertain as well.

Another problem is that there is no validation of results. For instance, Carter and Carter could have tested the algorithm on the simple case of spacing eight points in a cube to see if the optimal solution (a point in each corner) is obtained. There is not even a discussion as to the physical desirability of the colors that were generated by the algorithm.

Also, the computer code is poorly documented and difficult to understand. Variables are not identified, variable names are often conflicting and confusing, and statement labels are poorly numbered and sometimes not found. As for the documentation, the number of internal comments is insufficient to make the program easily understandable.

Carter and Carter's method has its good points as well. For example, it is easy to understand. Best of all, the technique apparently yields solutions that are at least close to optimal, which is all they really set out to do anyway. But, they could possibly do better. For instance, the two closest points may be optimally spaced, but what about the rest of them? Wouldn't it make sense to go through the algorithm once, then maximize the second closest distance, the third closest, and so on, until all the points are optimally spaced?

Also, Carter and Carter maneuver their points in one color space and then transform them to another color system to calculate the distances. I believe a more appropriate approach would be to do all the maneuvering and calculations in the same coordinate system. This would not only increase the efficiency of the program, it would increase its flexibility as well. It would enable the algorithm to solve any type of spacing problem, not just the color spacing problem. An algorithm like Carter and Carter's, but with better validation and documentation, including the concept of successively maximizing the minimum distance, second minimum distance, third minimum, and so on, as well as continuous operation in the same coordinate system, should make for an improved, more reliable, more understandable, more efficient, and more flexible solution technique.

IV. Mathematical Development

If you're afraid of mathematics, like so many of us are, this section might not appeal to you at first. But don't be frightened, because the mathematics here are really nothing more than algebra, although the terminology and notation might be somewhat difficult to keep track of. The first part will show the mathematical formulation of the color-spacing problem in general terms. Next, there will be an examination of some of the solutions to simple spacing problems. The final section will be devoted to discussing how the problem could be solved by nonlinear programming.

Problem Formulation

For the spacing problem, we want to maximize the minimum distance among n points in a convex polyhedron. It's a mouthful, isn't it? Let's take it apart and define some terms. A polyhedron can be simply defined as a region bounded by "many faces." A triangle is an example of a polyhedron that has three faces; a square has four faces. In three dimensions, a cube and a box are each polyhedrons with six faces while a pyramid has five. These are all convex regions as well, because there are no "juts" sticking into the region. A star (for instance, one of those on the U.S. flag) is not a convex region for that very reason. More formally, a region is convex if every point in that region can be connected to every other point in the region with a straight line that stays completely within the region. All it means is that no juts or dents are allowed.

In two dimensions, the region bounded by a circle is convex, as is that of a sphere in three dimensions, but neither one of them is a polyhedron because they have no faces. Mathematically speaking, a polyhedron must have linear boundaries. That means that a polyhedron must be made up of a series of straight lines (or planes if it's three-dimensional). No curved boundaries are allowed. Since the color spacing problem is a three-dimensional one, we will largely concern ourselves with three dimensions from now on.

The boundaries are known in the trade as constraints. A linear constraint is of the form:

$$ax + by + cz = k \tag{2}$$

where

(x,y,z) = the coordinates of a point

k = a constant known as the right hand side value

a, b, and c = any real numbers

The above equation describes a plane. Changing the equation to $ax + by + cz \leq k$, we have the points on a plane in addition to those on one side of it. But our region has many faces to it, so that there are actually many constraints of the form:

$$a_1x + b_1y + c_1z \leq k_1$$

$$a_2x + b_2y + c_2z \leq k_2$$

$$a_3x + b_3y + c_3z \leq k_3$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$a_fx + b_fy + c_fz \leq k_f$$

where

f = the number of faces which bound the region

Together, these constraints form a region within which all of our points must stay. If a point is outside of the region, then at least one of these constraints will be violated.

Let's turn our attention again to the objective. We want to maximize the minimum distance among n points. The distance from one point to another in the region can be measured by the Euclidean distance formula:

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \qquad (3)$$

where

$d(i,j)$ = the Euclidean distance from point i to point j

(x_i, y_i, z_i) = the coordinates of point i

In order to solve the problem, we need to find the minimum distance between all pairs of points. If there are only three points, it's easy;

there are only three pairs of points and we want the min $\{d(1,2), d(1,3), d(2,3)\}$ where min $\{\ \}$ denotes the smallest value of all the numbers in the set in brackets. But for n points, that set may be quite large. The minimum distance for n points would be:

$$\min \{d(1,2), d(1,3), d(1,4), \ldots, d(1,n), d(2,3), d(2,4), \ldots, d(2,n), d(3,4), \ldots, d(3,n), \ldots, d(n-2,n-1), d(n-2,n), d(n-1,n)\}$$

As a matter of fact, there are $n(n-1)/2$ values in the set. Letting $D = \min \{d(i,j)\}$, the objective then is to:

$$\text{maximize } D = \text{maximize } \min \{d(i,j)\} \tag{4}$$

subject to

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} = d(1,2)$$

$$\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2} = d(1,3)$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$\sqrt{(x_{n-1} - x_n)^2 + (y_{n-1} - y_n)^2 + (z_{n-1} - z_n)^2} = d(n-1, n)$$

$$a_1x_1 + b_1y_1 + c_1z_1 \leq k_1$$

$$a_1x_2 + b_1y_2 + c_1z_2 \leq k_1$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$a_1x_n + b_1y_n + c_1z_n \leq k_1$$

$$a_2x_1 + b_2y_1 + c_2z_1 \leq k_2$$

$$a_2x_2 + b_2y_2 + c_2z_2 \leq k_2$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$a_2x_n + b_2y_n + c_2z_n \leq k_2$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$a_fx_1 + b_fy_1 + c_fz_1 \leq k_f$$

$$a_fx_2 + b_fy_2 + c_fz_2 \leq k_f$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$a_fx_n + b_fy_n + c_fz_n \leq k_f$$

where

$i = 1$ to $n-1$

$j = i + 1$ to n

n = the number of points

f = the number of faces

the first set of constraints = the distance equations

the second set of constraints = the boundary constraints for all n points

We can eliminate the square root signs in the distance constraints and it will not affect the formulation, so let's do that just to uncomplicate the equations. Also, we can subtract D from all the distance constraints to create equations of the form:

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D = d(i,j) - D \qquad (5)$$

But we know that $d(i,j) - D$ is going to be a number greater than or equal to zero because D represents the smallest possible value of any $d(i,j)$. Therefore, we can change the equation above to:

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D \geq 0 \qquad (6)$$

The problem now becomes:

$$\text{maximize } D \qquad (7)$$

subject to

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D \geq 0$$

$$a_1 x_g + b_1 y_g + c_1 z_g \leq k_1$$

$$a_2 x_g + b_2 y_g + c_2 z_g \leq k_2$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots$$

$$a_f x_g + b_f y_g + c_f z_g \leq k_f$$

where

$i = 1$ to $n - 1$

$j = i + 1$ to n

g = a subscript for the 1 through n points that must stay within each boundary constraint

n = the number of points

f = the number of faces

At this point you might think we're done, but there are a few more simplifications to go. For practically all commercial methods of constrained optimization, the constraints are required to have equal signs instead of inequalities. We accomplish this by using variables that are commonly referred to as surplus and slack variables. For example, we know from before that the value of $d(i,j) - D$ is non-negative, but we don't know how large it is. If we call this value s_{ij} and subtract it from the distance constraint, we have:

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D - s_{ij} = 0 \qquad (8)$$

The constraint is now an equality constraint as required, and s_{ij} is called the surplus variable because it represents how much greater than zero equation (6) is.

Similarly, we can add what is called a positive slack variable to the boundary constraints to make equations of the form:

$$a_h x_g + b_h y_g + c_h z_g + S_{hg} = k_h \tag{9}$$

where

S_{hg} = the slack variable

The problem can then be formulated as:

$$\text{maximize } D \tag{10}$$

subject to

$$(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D - s_{ij} = 0$$

$$a_h x_g + b_h y_g + c_h z_g + S_{hg} = k_h$$

$$x_g,\ y_g,\ z_g,\ D,\ s_{ij},\ S_{hg},\ k_h \geq 0$$

where

$i = 1$ to $n - 1$

$j = i + 1$ to n

g = a subscript for the 1 through n points

h = a subscript for the 1 through f boundary constraints

n = the number of points

f = the number of faces

s_{ij} = surplus variables for the distance constraints

S_{hg} = slack variables for the boundary constraints

The third set of constraints states that x, y, z, D, s, S, and k must all be greater than or equal to zero, commonly referred to in the business as the non-negativity constraints. This is another requirement of the commercial optimization techniques. Let's talk about the non-negativity constraints. We already know that D is positive (you can't have a negative distance), and s_{ij} and S_{hg} are also positive as we discussed before. If the x's, y's, and z's could possibly come out negative, we could make transformations of the form $x_i' = x_i - t$ where t is the minimum possible value of x, to ensure that x_i' is positive. If k is negative, then you can multiply both sides of the equation by -1 to alleviate the problem, but the sign changes from less than to greater than, and S_{hg} becomes a surplus variable instead of a slack variable. These steps ensure that S would always be positive as well as k, and the non-negativity constraints are now completely satisfied.

Notice that we now have $n(n-1)/2$ distance constraints that are nonlinear, and $n \cdot f$ linear boundary constraints. The next step is to incorporate the nonlinear constraints into the objective function with what is called a Lagrangian function. If we call the left side of the first distance constraint Q_{12}, the second Q_{13}, and so on, the Lagrangian would look like this:

$$L = D + \lambda_{12}Q_{12} + \lambda_{13}Q_{13} + \dots + \lambda_{1n}Q_{1n} + \dots + \lambda_{n-1n}Q_{n-1n} \qquad (11)$$

or similarly:

$$L = D + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \lambda_{ij} \left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D - s_{ij}\right] \tag{12}$$

where

L = the Lagrangian function

λ_{ij} = the Lagrangian variables

So, the formulation of the problem now looks like this:

maximize

$$L = D + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \lambda_{ij} \left[(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 - D - s_{ij}\right] \tag{13}$$

subject to

$$a_h x_g + b_h y_g + c_h z_g + S_{hg} = k_h$$

$$x, y, z, D, s, S, k \geq 0$$

where

g = a subscript for the 1 through n points

h = a subscript for the 1 through f boundary constraints

i = 1 to $n - 1$

j = $i + 1$ to n

n = the number of points

f = the number of faces

s_{ij} = surplus variables for the distance constraints

S_{hg} = slack variables for the boundary constraints

(x_g, y_g, z_g) = coordinates of point g

λ_{ij} = Lagrangian variables

And that's it. We now have a nonlinear objective function and $n \cdot f$ linear constraints. Additionally, the objective function is mathematically classified as convex because it is the summation of many convex functions. This is somewhat unfortunate because if it were concave, the problem would fall under a category of problems which could be solved using convex programming; a solution technique which is widely used in operations research. Before we leave this section, I would like to mention one more thing. Originally, it was thought that the color spacing problem had completely linear boundary constraints (i.e., that the region was a polyhedron). It turns out that this is not the case. It is indeed a convex polyhedron in the red, green, blue coordinate system, but in the one that counts, the CIE L*u*v* system where distances are computed, it is not. However, it is still convex and the general formulation above still holds. Those constraints that are nonlinear can be moved into the Lagrangian function in much the same manner as the distance constraints were, while the linear boundary constraints can stay put. That concludes the problem formulation, and we are now ready to look at the solutions to some easy spacing problems.

Solutions to Simple Problems

This is the section where the puzzle is solved. Although the solutions here are for very simple problems and many of them can be recognized intuitively without any trouble, some of them are not so apparent. What I'm going to show you here are the solutions for spacing 2, 3, 4, 5, and 6 points in a square to get a feeling for the types of solutions that can be expected from the spacing problem. Included is a discussion of the mathematical aspects of some of the solutions and some conclusions based on what we've seen.

Taking a look first at the solution for two points (shown in Figure 4), we can see that it is exactly what one would expect. The points are in opposite corners. Realize also that there are actually two possible optimal solutions that will yield the same value for D. One is pictured here and the other would, of course, have the points in the other two corners.

The solution for spacing three points is somewhat less obvious (see Figure 5). It turns out that the points form an equilateral triangle

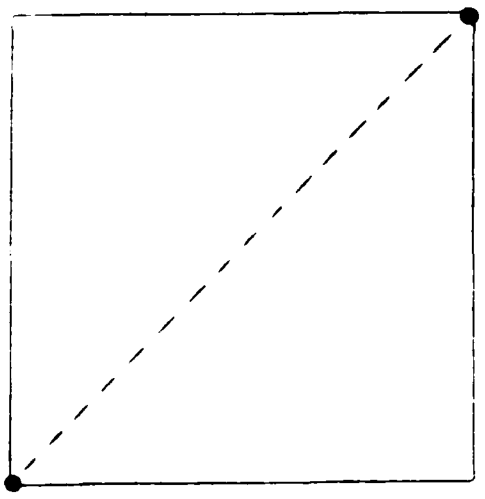


Figure 4. Solution for Spacing Two Points in a Square

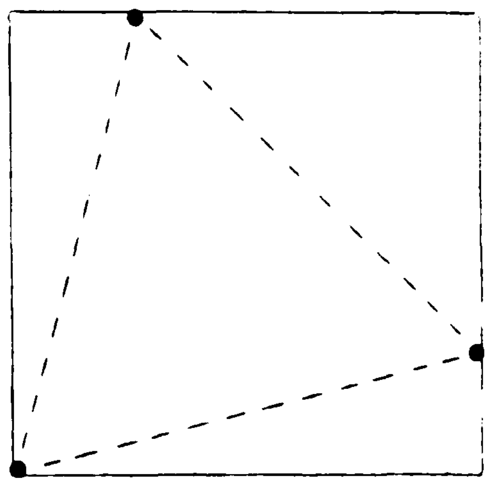


Figure 5. Solution for Spacing Three Points in a Square

(i.e., a triangle whose sides are of equal length) with one point in a corner and the other two points on opposite faces at angles of 15 degrees from the corner point. And, not coincidentally, their coordinates are (0,0), (1, tan 15 degrees) and (tan 15 degrees, 1), respectively, for a square whose sides are one unit long. In addition, one can easily tell that this problem actually has four solutions that are multiple optimal; each one corresponding to a different corner point.

Spacing four and five points in a square is trivial. For four, the points belong in the corners. The five-point solution is identical to the four-point solution with the additional point going in the center of the region (see Figures 6 and 7). Incidentally, both of these problems only have the one optimal solution.

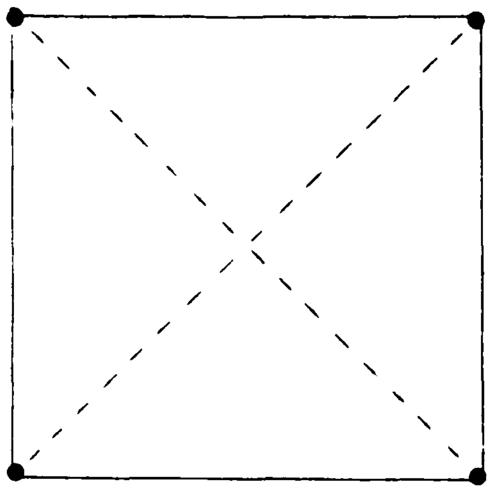


Figure 6. Solution for Spacing Four Points in a Square

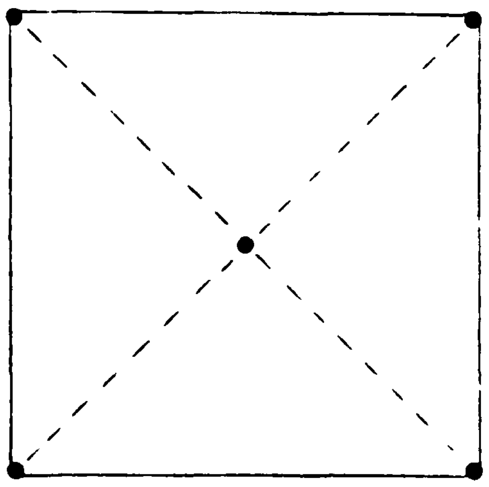


Figure 7. Solution for Spacing Five Points in a Square

Now we get to the really exotic one. The optimal solution for spacing six points in a square is shown in Figure 8. Is that what you would have expected the solution to look like? If you're like most people, probably not. I would have guessed that all four corner points would be filled and that the two other points would be somewhere in the interior. Instead, there are two corner points, one interior point, and three points on the faces. The presence of a counterintuitive solution like this one indicates that accepting human judgment for good solutions may be risky. It further emphasizes the need for a computer generated solution technique like the one introduced here. Notice that the left two-thirds of the region is identical to a skinny five-point solution, and that the points in the right two-thirds form a diamond where all the sides are of equal length. In all, there are six distances that are identical--the four of the diamond and the two connecting the interior point to the corner points--and they all correspond to the *minimum* distance. Notice also that there are four multiple optimal solutions for this problem.

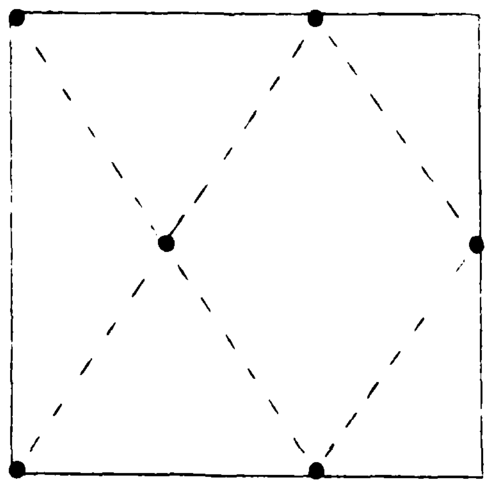


Figure 8. Solution for Spacing Six Points in a Square

That brings us to the conclusions that can be drawn from these simple examples. First, it appears that the points generally tend to fall in the corners and the faces before they go to the interior of the region. Second, the solutions are not always what one would expect. The six-point problem yields a good example of that phenomenon. Third, most of the problems have multiple optimal solutions, so that the decision maker can often choose between several alternatives. Finally, it appears that the optimal solutions all have a high percentage of their distances identical, and equal to the maximized minimum distance. For instance, the three-point solution has 3 out of the 3 equal to D, the four-point solution has 4 out of 6, the five-point has 4 out of 10, and the six-point has 6 out of 15. This means that one might be able to tell how good his or her solution is not only by the value of D, but also by the number of distances that are equal to that value.

Solution by Nonlinear Programming

Up to now, you may have been thinking that the problem looks interesting, but the mathematical formulation appears pretty complex. How does one actually solve it? Well, there are several ways. You can write a heuristic algorithm like Carter and Carter did, and like I did. Or you can solve it intuitively if it's simple enough. Or you might be able to use a computerized nonlinear optimization technique. This section will discuss one such nonlinear programming method.

It was originally called the Method of Approximation Programming (MAP) when it was first introduced back in the late 1950s by Griffith and Stewart (7), but now it is perhaps better known as successive linear programming (SLP). It works like this: Each nonlinear constraint is linearized using a first-order Taylor's series expansion. For those of you who don't know, the Taylor's series expansion of an equation is just a mathematical series of summed terms which approximates the original equation and converges to the original with enough terms in the series. Once the nonlinear constraints are linearized, the user selects a starting set of values and solves the now completely linear problem using linear programming (LP). The solution to that linear programming problem is then used as the new starting set, and another LP solution is generated. This goes on until the LP solutions become identical, at which time the LP solution is also the solution to the nonlinear problem.

There are certain restrictions on the type of problem for which this technique will guarantee an optimal solution, yet Griffith and Stewart state that "problems have been solved with MAP which do not fully satisfy all of these requirements" (7:379). The reason this technique was not chosen to solve the color spacing problem is because of the size of the problem. For example, in order to solve Carter and Carter's problem for 25 points, there would be 175 boundary constraints (some linear, some nonlinear), 300 nonlinear distance constraints, and 76 decision variables. It is uncertain whether an LP program could solve that problem in a reasonable amount of time, let alone solve it again for the second closest distance, the third closest, and so on, as the heuristic algorithm introduced in this thesis is able to do. Although this problem may not seem like an impossible task for some commercial LPs, the primary objective of this effort is to provide AFAMRL with a technique that works. For that reason, I chose to solve the problem with a heuristic algorithm instead of SLP.

V. Description of Heuristic

If you're reading through this section and it looks vaguely familiar to you, it's not a case of precognition on your part, so don't call the psychiatrist. It's because the heuristic described here is largely patterned after that of Carter and Carter which was described in Section III. However, there are some major and very important differences, so don't think that you can skip to the next section because you already know all about it. On the contrary, you should read through this entire section if for no other reason than to please me. I will be extremely pleased if you read the first portion which basically describes how the algorithm works and some of the logic behind it. And, I will be ecstatic if you read the second portion which identifies the differences between this algorithm and Carter and Carter's.

The Algorithm

Recall that the purpose of this thesis is to maximize the minimum distance among *n* points in a convex region, and then as a kicker to successively maximize the second minimum distance, the third minimum, and so on, until all points are optimally spaced. The first step in achieving this objective is to randomly place the points uniformly throughout the region. For the color spacing problem, it would look like the one shown in Figure 9. Recall from before that this region is not a polyhedron, although Figure 9 depicts it as one. The actual color spacing region would have many of its boundaries bowed out slightly. Because they are

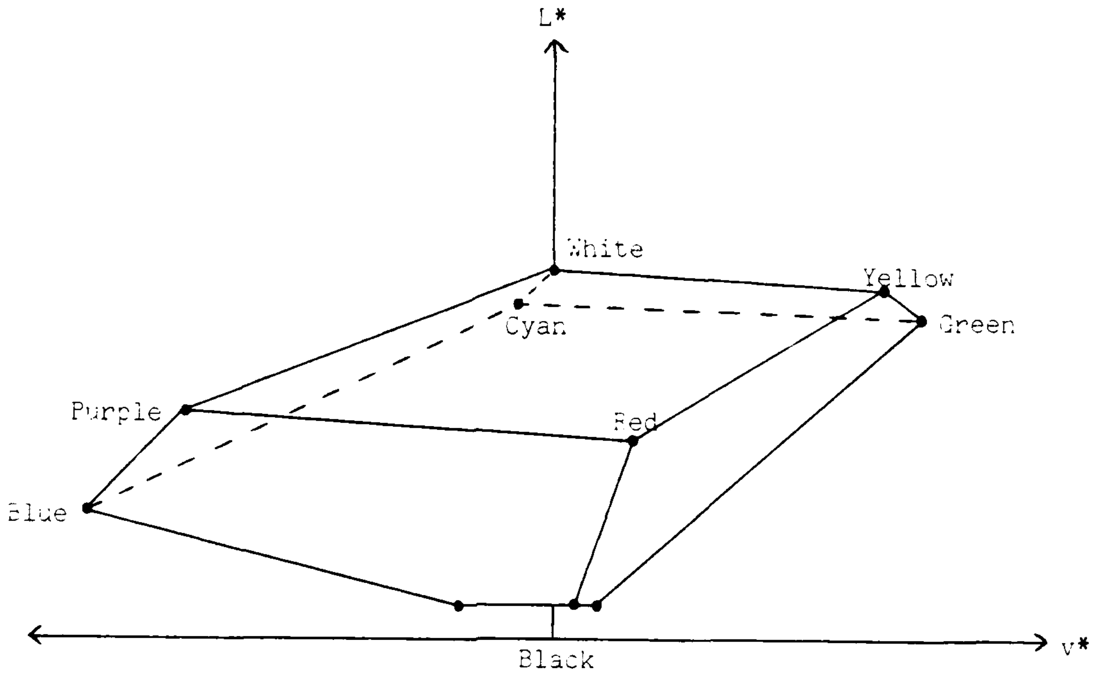


Figure 9. Approximate Feasible Region for the Color Spacing Problem in the CIE L*u*v* Color System

bowed out and not in, the region is still convex, so the solution technique described here still applies.

In step 2, the program finds the minimum distance among all n(n-1)/2 pairs of points, and the endpoints that correspond to that distance. The third step is to define the 27 alternative locations for each of the two endpoints found in step 2. These alternatives lie on a unit cube around the endpoint whose sides are twice the step-size in length. Figure 3 on page 9 depicts these alternative endpoint locations.

In the fourth step, the alternatives are checked to see if an improvement can be made by moving the endpoints. The program does this by first checking the alternatives to see if they are in the region. Then, taking one legal alternative at a time, it checks the discance between endpoints to see if that value is greater than before. Next, it checks the new overall minimum distance to see if that is greater. If all of those conditions are successfully met, then the endpoints are moved and the routine checks the next alternative for an even greater possible improvement. It does this for every possible combination of alternative endpoint locations; a total of 729 combinations. If several alternatives have the same overall minimum distance, which is likely, the tie goes to the alternative producing the greatest distance between endpoints.

The program then returns to step 2 and finds the new overall minimum distance and the endpoints that correspond to that distance. This loop continues until no improvement can be made with the given step-size. At that time, the step-size is halved and the program begins looping again starting at step 2. This halving process continues until the step-size reaches some minimum value chosen by the user and we enter step 5.

The fifth step controls the fixing of points already maximized and the status of the program. When the step-size reaches its minimum tolerance, one of the endpoints is fixed in its position, the step-size is returned to its original value, and the program returns to step 2. The only difference in the logic now (other than a point being fixed) is that the minimum distance calculated in step 2 can't have two fixed points as its endpoints. This enables the program to concentrate on the second shortest distance. The fixing of points continues until all the points become fixed, at which time they are all unfixed and the routine starts all over again at step 2 as if the current location scheme were the starting positions. If the program runs all the way through again and the points haven't moved, then the program is done. Otherwise, the program keeps going until there is no change in location for one complete iteration. Figure 10 contains a flowchart of the program's logic, while Appendix C contains an example of how the program optimally spaced four points in a square.

Differences with Carter and Carter's Method

As I said at the top of this section, there is a striking resemblance between Carter and Carter's method and the one described here. However, there are four major differences which I would like to discuss at this time.

The first of these differences lies in the checking of alternatives to see what expanding move should be made. Carter and Carter move only one endpoint at a time and, therefore, choose between only 52 possible endpoint location schemes (26 for each endpoint). On the other hand, the Roley algorithm moves both endpoints at once and, as a result, has a total of 729 possible endpoint location schemes. It's just a more exhaustive search for the best possible move, that's all.

Along the same lines, Carter and Carter's method ranks the alternatives as to how much they increase the distance between endpoints and then chooses the highest ranking alternative that also increases the overall minimum distance. On the other hand, my program chooses the alternative that most increases the overall minimum distance with ties going to the alternative which causes the greatest increase in the endpoint distance. The difference is that Carter and Carter's algorithm may

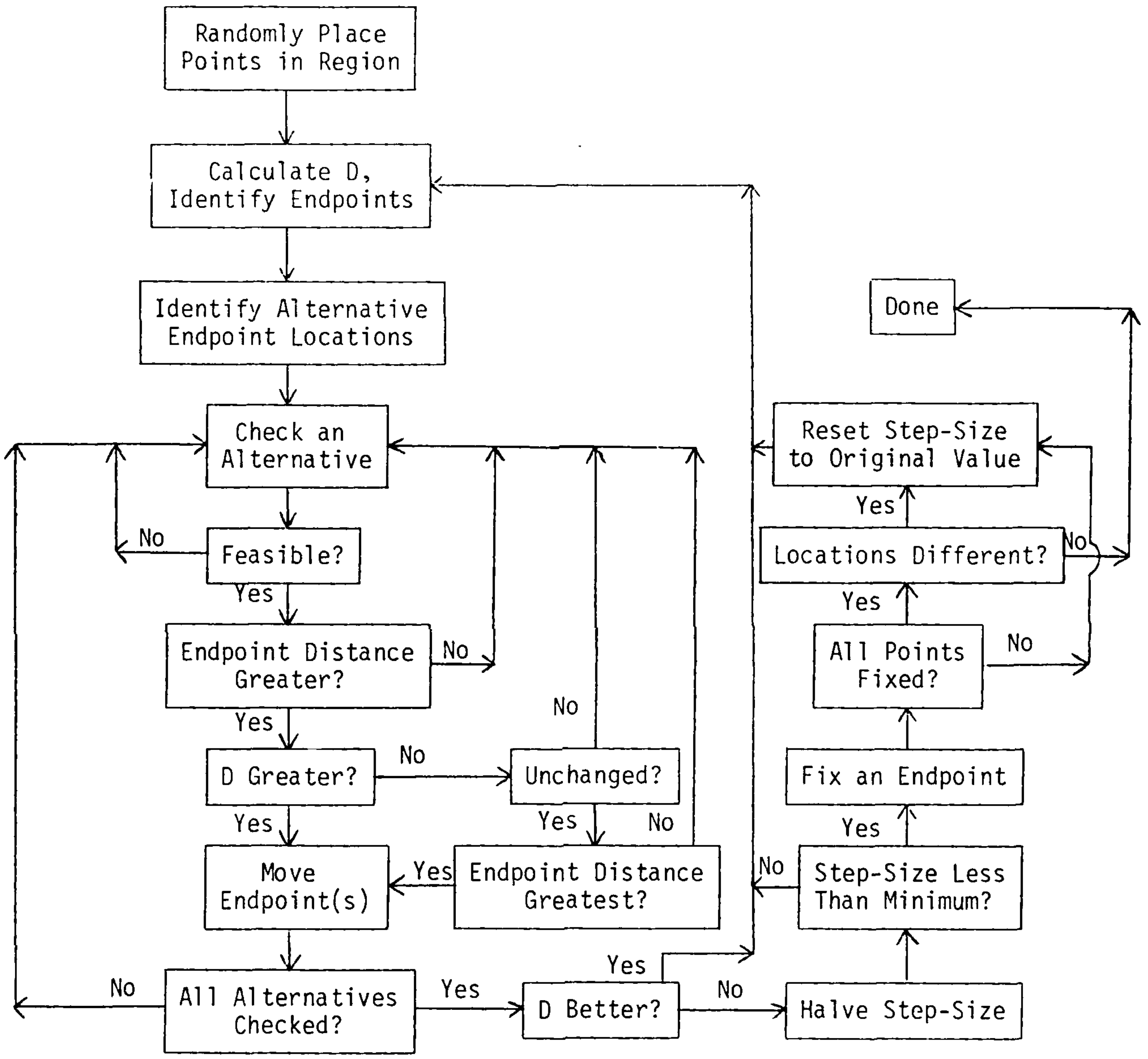

Randomly Place Points in Region
Calculate D, Identify Endpoints
Identify Alternative Endpoint Locations
Check an Alternative
Feasible?
No
Yes
Endpoint Distance Greater?
No
Yes
D Greater?
No
Yes
Unchanged?
No
Yes
No
Move Endpoint(s)
Yes
Endpoint Distance Greatest?
All Alternatives Checked?
No
Yes
D Better?
Yes
No
Halve Step-Size
Step-Size Less Than Minimum?
No
Yes
Fix an Endpoint
All Points Fixed?
No
Yes
Locations Different?
No
Yes
Reset Step-Size to Original Value
Done

Figure 10. Flowchart of Roley Heuristic

not choose the best alternative for increasing the overall minimum distance, whereas mine does.

Another difference has to do with the thoroughness of the spacing method. Carter and Carter maximize the minimum distance but leave the second minimum, third minimum, and all the others where they are. My algorithm not only maximizes the minimum distance, it successively maximizes the second minimum distance, the third minimum, and so on until all points are optimally spaced. The result is a complete optimization of all distances, not just the minimum distance. The price is a more complex and slower computer algorithm.

The fourth and final difference has to do with the respective coordinate systems that are used as the primary operating spaces. Carter and Carter's technique spaces points in the red, green, and blue color system, and converts the colors to CIE L*u*v* color coordinates in order to calculate spatial distances. My program acts in a reverse manner. It maneuvers points and calculates distances in the CIE L*u*v* system, but checks color locations in the red, green, and blue system to make sure thay are within the boundaries of the region. It was originally hoped that all operations and calculations could be done in the CIE L*u*v* system for increased efficiency, but the equations for the boundary constraints could not be derived in the L*u*v* system, preventing its use as a location checking coordinate system.

VI. Results

Now that you know how the thing works, let's find out how well it works. There are several questions that need to be answered in this section. Does the algorithm guarantee an optimal solution? Does it work for very large problems? How well does it perform in comparison to Carter and Carter's method? What are some of the factors to which the program is sensitive? What are its biggest advantages? What are its biggest disadvantages? These are the questions that will indeed be answered in this section.

Optimality

The algorithm does not guarantee an optimal solution. However, it was tested on a number of problems whose solutions are known and the results are encouraging. These problems include spacing three, four, five, and six points in a square and spacing eight points in a cube. Each problem was tested with three different sets of randomly placed points and the algorithm produced the optimal solution in 14 of the 15 problems. That's a success rate of 93 percent. The only unsuccessful attempt involved one of the tries at spacing five points in a square. In that instance, the minimum distance arrived at was .638 units compared with the optimal solution of .707, so that it was within 90 percent of optimal. In general, the more replications that are performed, the better chance of success.

Size Limitations

This may be the largest downfall of my program. I had originally intended for it to accommodate up to 50 points and 10 faces. However, the largest problem it has been able to successfully solve was the color spacing problem with 23 colors and 7 faces. Larger problems exceeded the time limit of 1000 CPU seconds on the CDC 845 (Cyber) computer. The reason for this size deficiency boils down to the thoroughness of the program. Each iteration checks $n(n-1)/2$ distances for up to 729 alternatives. It takes 21 iterations for the step-size to reach its minimum value. The program must perform those 21 iterations for each of the n points that are fixed in succession. It then goes through the whole process at least one more time or until the point locations do not change. Experience has shown that it usually goes through twice. With an increase in n, the number of computations (and thus the CPU time required) goes up exponentially, so one can easily tell that size is a serious factor. Presumably, reducing the number of alternatives, increasing the minimum step-size, or reducing the amount of point fixing would increase the ability of the program to handle larger problems.

Performance in Comparison to Carter and Carter's Method

The comparison of my program with Carter and Carter's was done with identical color parameters of Y_0, u'_0, v'_0 , and the chromaticity coordinates of the guns. Table I contains comparative results for spacing 3, 4, 6, 10, and 25 colors.

TABLE I.

COMPARISON OF MINIMUM DISTANCES

					Average Computer Time Per Replication	
Number of Colors	Method	Best Answer (CIE L*u*v*)	Variance	Number of Repli-cations	CDC 845 (sec)	UNIVAC 1182 (sec)
3	Roley	239.26	1.17	5	23.50	
	Carter	239.33	.11	49		1.31
4	Roley	156.28	136.62	5	42.95	
	Carter	155.87	37.14	50		1.76
6	Roley	124.85	79.61	5	100.49	
	Carter	124.08	62.45	50		2.48
10	Roley	71.07	48.61	20	178.90	
	Carter	89.43	27.73	50		4.22
25	Roley	0.00	0.00	5	>1000.00	
	Carter	51.60	7.34	46		12.39

As an explanation, I originally intended to perform only five replications for each problem because time considerations prevented me from performing 50 runs as Carter and Carter did. However, I ended up doing 20 runs of the ten color problem to try to decrease the great disparity of results. Also, I was unable to successfully execute the 25 color problem because the CPU limit of 1000 seconds was exceeded for each attempt.

It is difficult to compare this algorithm with Carter and Carter's because there are so many factors that can be used as a measure of effectiveness. Most people, however, would consider the minimum distances produced by each method as their primary concern. Notice in Table I that the minimum distances are very similar for spacing 3, 4, and 6 colors although Carter and Carter's results are based on approximately 10 times more runs. The case of spacing 10 colors is perplexing. One would assume that the results would again be equivalent because the techniques are so similar. Apparently, that is not true. My heart tells me that something is wrong, but it is difficult to confirm Carter and Carter's results because those actual color locations were not published and are not available at this time. Perhaps an even greater number of replications would yield a better solution. In the meantime, I can only admit that my algorithm is deficient for spacing larger numbers of colors.

Another criteria one might use to judge the two methods would be to compare the distances between the second closest points, third closest, and so on. This would indeed be an appropriate measure because the fundamental conceptual difference between the two methods is that mine concentrates on successively maximizing all distances, whereas Carter and Carter's maximizes only the minimum distance. Presumably, these efforts should have paid off, but it is difficult to judge because of a limited sample size. Table II presents a comparison of distances for the six-color spacing problem. I would have liked to compare other problems in addition to the six-color one but it was the only one that was published and available, so the sample size isn't quite what one would need to make a definitive judgment. Notice, however, that the first three distances are slightly better for my algorithm, the next six are fairly even, while the last six are clearly in Carcer's favor. This could indicate that, while maximizing the second and third minimum distances produces better results for those values, it may cost in the long run in the form of smaller values for the greater distances. Notice also that the first

TABLE II.

COMPARISON OF ALL DISTANCES FOR THE SIX-COLOR PROBLEM

ith Minimum Distance	Roley	Carter	ith Minimum Distance	Roley	Carter	ith Minimum Distance	Roley	Carter
1st	125*	124	6th	129	130*	11th	182	218*
2nd	125*	124	7th	136*	133	12th	201	237*
3rd	125*	124	8th	138	140*	13th	220	239*
4th	125	125	9th	149*	147	14th	239	249*
5th	125	125	10th	171	210*	15th	241	263*

*Denotes the superior value.

through sixth distances are basically the same regardless of whether or not they are maximized. They seem to have been automatically maximized by maximizing the minimum distance. That indicates that successively maximizing higher distances may not be worth the extra computer time required. These results also support the observation that was made in Section IV that optimal solutions will generally have several distances equal to the minimum distance. It is a characteristic that could be extremely useful in determining whether or not a particular solution is close to optimal.

One might also be impressed with my method if it could take Carter and Carter's best solution for a particular problem, use it as a starting location scheme, and improve upon it. Using Carter and Carter's six-color solution (because it was the only one available) as an initial set of colors resulted in a mild improvement of the minimum distance from 124.026 CIE L*u*v* units to 124.145 units. Research by Carter and Carter has shown that distinguishability between colors "deteriorates rapidly when the distance between colors is about 40 CIE L*u*v* units," (2:2937) so this slight improvement of .12 unit is inconsequential. Perhaps more importantly, my algorithm improved on 11 out of the 14 other distances with one unchanged and two decreasing slightly. You may have noticed that the original distance of 124.026 does not correspond to Carter and Carter's calculated value of 124.08 as shown in Table I. Taking their same set of color locations in red, green, and blue coordinates, I calculated a minimum distance of 124.026 in L*u*v* coordinates instead of 124.08 (the exact L*u*v* coordinates were not published). The difference is probably due to roundoff error. But, how does one explain how I could improve on their optimal solution with basically the same technique? Accuracy. Because Carter and Carter's minimum step-size is one unit, they are not able to attain the same level of accuracy as my heuristic does with a minimum step-size of .0001 units. Consequently, the Carters are not able to "snuggle" their points into the best locations. All this indicates is that the minimum step-size is a control over the level of accuracy one wishes to achieve.

As was briefly discussed before, the question of computer time may also be an important performance characteristic by which to judge the two methods. Theoretically, Carter and Carter's method should be considerably faster because it maximizes only the minimum distance and isn't concerned with the second minimum distance, third minimum, and so on. Looking again at Table I, we can see that this is, in fact, the case. It should be remembered, however, that comparing run times on two different computers is somewhat like comparing apples and oranges and should be taken with a grain of salt.

As far as those at AFAMRL and I are concerned, the chief criterion for comparison is which program works. I personally know of no place where Carter and Carter's algorithm is currently running and several places where people have tried to implement it and have not been able to, AFAMRL included. On the other hand, my program is one for one so far so it must be given the edge. Interested parties will be delighted to find out that my program is heavily documented internally, easy to understand, utilizing classic structured programming concepts and written in FORTRAN 77. These characteristics give my program the added advantage of being easier to modify and/or upgrade by others who may be so inclined. A listing of the code is contained in Appendix B for all to judge for themselves.

Briefly summarizing this section, we found that Carter and Carter's method works equally as well for finding the maximum minimum distance, but requires more runs. For the six-color spacing problem, theirs yielded slightly worse values for the second and third minimums, and much better results for the greater distances. Carter and Carter's program takes less time, but has a slightly lower degree of accuracy. Unlike Carter and Carter's, my program works, is well documented, and easy to understand.

Sensitivity to Parameters

The program introduced in this report is highly sensitive to four parameters. They are the maximum step-size, the step-size reducing scheme, the minimum step-size, and the initial random positioning. The maximum step-size is important because some solutions are not obtainable for certain step-sizes and point locations. Take, for instance, the case of spacing eight points in a cube whose sides are one unit long. Suppose seven of the points are in the corners and the eighth is in the very middle of the cube. The eighth point should migrate to the empty corner, but unless the step size is greater than one-third of a unit, the minimum distance will decrease if it tries to move in any direction. Essentially, the point is "locked in" to its location. To avoid this problem, I recommend using a maximum step-size equal to approximately half the distance between the two farthest points in the region. This way, a point located in the middle can move to any other spot in the region.

Granted, a point doesn't necessarily have to be in the middle to be locked in. Let's say the eighth point in our example above is not located in the exact middle. Let's say instead that it is located towards the empty corner but less than one-third of a unit from the middle. A step-size of .5 now puts the point out of the region, so it still can't move closer to where it's supposed to be. If the step-size is halved, the point still may not be able to go towards the corner. But, some other step-size reduction scheme like reducing it by four-fifths might send the point to the optimal solution. In that sense, the reduction scheme is another factor which could affect the solution. The problem with that alternative is that it takes quite a few more iterations. For instance, to go from .5 to .0001 by halves, it takes 13 iterations. To go from .5 to .0001 by four-fifths takes 39 iterations. More iterations means more computation time and, thus, a less desirable program. The Carters and I both used a halving scheme, but I would recommend that any user experiment with this parameter to find out what is best for his or her application.

The minimum step-size, although a factor to the solution, is not as important as the two parameters discussed above. The minimum step-size controls the accuracy of the solution. For three decimal places of accuracy, .0001 should be the minimum step-size. For two places, use .001, and so on. Increasing the minimum step-size can cause a surprising decrease in the number of iterations required. For instance, it takes only six iterations to go from .5 to .01 by halves as opposed to 13 iterations to go to .0001.

The final parameter that significantly affects the solution is the seed. This is the starting value that is used in the random number generating sequence which calculates the starting locations for the points. One can intuitively realize the importance of the seed by understanding that some initial location sets are better than others. A bad location set is shown in Figure 11. None of the points in that example

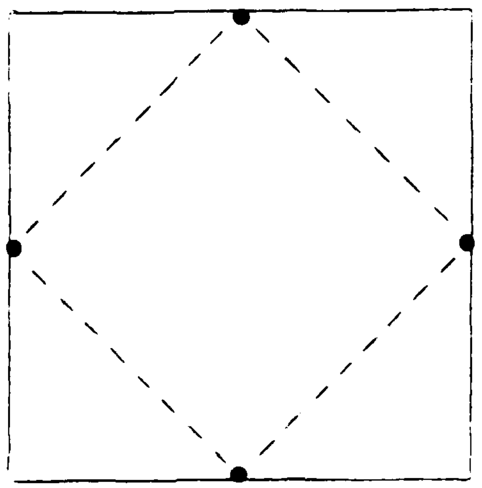


Figure 11. Example of Locally Optimal Solution

can move anywhere without getting closer to another point, and yet the solution is not optimal. It is called a "locally" optimal solution as opposed to the "globally" optimal solution which has a point in each corner. Spacing problems apparently tend to have many local optima. Changing the seed and doing numerous replications will increase the chances of finding a global optimum.

Advantages

The heuristic algorithm described here has numerous advantages. Here is a list of some of them:

1. Flexibility. The program works for any spacing problem including the color spacing problem. It has a myriad of uses in the field of optics, as described by Carter and Carter (2:2939). It can even be extended to four dimensions for possible use in the field of physics. It can accommodate distance formulas other than the Euclidean distance formula or convex regions besides polyhedrons. Its uses are limited only by the limits of the imagination.

2. Simplicity. It is simple to understand and simple to use. The structured FORTRAN programming format makes it simple to modify or debug.

3. Reliability. It has been tested favorably against simple problems with known solutions and more complex problems like Carter's color spacing problems. It has proven to produce good

answers with fewer replications than previous solution techniques.

4. Successive Maximization. It not only maximizes the minimum distance, it successively maximizes the second minimum, third minimum, and so on. Because all points and their distances in relation to each other are important in the spacing problem, this is a more appropriate objective.

Disadvantages

Here is a list of the method's major disadvantages:

1. Optimality. The method does not guarantee a globally optimal solution, nor does it guarantee even a locally optimal solution. It just guarantees a "good" answer and it may take several runs to get that "good" answer.

2. Size Limitations. The program has been proven on a problem of at most 23 points and seven faces. There is no guarantee that it will work on very large problems. Computer time is the key resource here because computer time required increases exponentially as the number of points increase. It's up to the user to weigh the value of his or her computer time to the value of a solution.

VII. Summary

I hope you've enjoyed this little trip into the world of polyhedrons, colors, and the like. If you've already forgotten what we've learned, here's a quick refresher. After being introduced to the spacing problem, we learned of some examples in the real world, most notably the color spacing problem. We were educated on the theory behind the color spacing problem and we found out how Carter and Carter proposed to solve it. Next, we learned how the spacing problem is formulated mathematically and were able to see some solutions to simple problems. We also found out how the problem could be solved using successive linear programming. Next, we were introduced to a new heuristic algorithm, very similar to Carter and Carter's, that is designed to solve the spacing problem. Finally, we found out how well the new algorithm works. It is now time to identify our conclusions and some suggestions for further research.

Conclusions

The program works. That was the main objective of this effort so in that sense, I have accomplished what I set out to do. Perhaps equal to that was the objective of providing AFAMRL with reasonable answers to the color spacing problem that are better than educated guesses. I believe that the algorithm does that as well. An objective of my research also was to improve upon Carter and Carter's method. Well, it is difficult to compare the two because time constraints prevented me from doing a full statistical analysis. But, if you consider that my method runs, is easy to understand and use, and yields solutions that are comparable to Carter and Carter's, one must consider it an improvement.

To be honest with you, I originally thought that Carter and Carter's algorithm was a crude first attempt generating suboptimal solutions which could easily be improved upon. It is a credit to their work that those solutions are apparently not so crude. But it must be remembered that regardless of which method is used, they both require numerous replications to come up with an answer that is only guaranteed to be "good." So there is still plenty of room for improvement.

Suggestions for Further Research

There are two directions that subsequent research in the area can take. One is to refine the new algorithm. The other, and most exciting alternative, is to solve the spacing problem using some form of nonlinear programming. The method described in Section IV of this report has some definite possibilities, but the difficulty is that the constraints of the color region are not linear and are very difficult to derive mathematically. I could not solve for them during my work on this thesis. Carter and Carter describe the region as "approximately a triangular prism," but nowhere in the literature is there a mathematical formulation of the boundaries (2:2937). This could prove to be a major stumbling block for anyone attempting to solve the color spacing problem using nonlinear programming.

Alternatively, there are numerous improvements that could be made to my new algorithm as well. The most worthwhile appears to be the use of vectors and matrices to help define possible new positions for the two closest points. The present method disregards any alternative that takes a point out of the feasible region. A method could be devised that would instead take the point to the boundary of the feasible region and still consider it as a viable alternative. Indeed, it may be a desirable move because of the tendency of the points to migrate to the boundaries and corners of the region. The trouble with this method is that it too only works for linear constraints. Colors would have to be spaced in the red, green, and blue coordinate system, which has linear boundaries, and then transformed into L*u*v* coordinates to calculate the distances, much in the manner that Carter and Carter's algorithm does.

Another interesting improvement would be to use a concept called "simulated annealing" to help space the points. In this method, the points would be able to actually move closer together in order to eventually achieve optimal spacing. Refer to Kirkpatrick, Gelatt, and Vecchi (9) for more on this possibility.

Other less drastic refinements might be to make the program interactive, more user-friendly, or perhaps include a color-naming subroutine that would give the user descriptive names of the colors selected. Also, improvements might be made to the speed and efficiency of the program. I have thought of a three-phase system of varying the maximum and minimum step-sizes and the step-size reducing scheme to hopefully get consistently better answers more quickly. It would consist of starting out with large maximum and minimum step-sizes and halving the step-size at each iteration. This would place the points in a roughly spaced configuration. The next phase would have a much smaller maximum step-size, reducing by four-fifths to the same minimum step-size. This would refine the points to what one would hope to be a rough global optimum location scheme. Phase 3 would concentrate on further refining the locations and the accuracy with a very small maximum step-size and a miniscule minimum step-size. The composite result would hopefully be impressive.

Yet another improvement would be to selectively place the points in the corners of the region then place any remaining points randomly on the faces, rather than randomly placing all points in the interior of the region as an initial location scheme. This procedure would conform more closely to the observations made in Section IV regarding simple problems.

The biggest deficiency of this thesis effort was that the algorithm was never fully tested statistically to properly compare its results with those of Carter and Carter. It is highly recommended that any future effort include extensive tests of this nature. I have found the topic to be interesting, important, and satisfying, and am sure that anyone else who pursues research in the area will gain similar satisfaction.

Appendix A. Color Transformations

This appendix briefly describes the transformations from red, green, and blue color coordinates to the CIE L*u*v* color space. If we let (Y_R, Y_G, Y_B) be the luminances for the CRT's red, green, and blue guns, respectively, then the first task is to find something called the tristimulus values (X_T, Y_T, Z_T). This transformation is given by the following matrix operation:

$$\begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix} = \begin{bmatrix} x_R/y_R & x_G/y_G & x_B/y_B \\ 1 & 1 & 1 \\ z_R/y_R & z_G/y_G & z_B/y_B \end{bmatrix} \begin{bmatrix} Y_R \\ Y_G \\ Y_B \end{bmatrix}$$

where

(x_R, y_R, z_R) = the chromaticity coordinates of the red gun

(x_G, y_G, z_G) = the chromaticity coordinates of the green gun

(x_B, y_B, z_B) = the chromaticity coordinates of the blue gun

These chromaticity coordinates correspond to the locations on a chromaticity diagram of the red, green, and blue colors corresponding to that particular cathode-ray tube gun, and are values between zero and one. In addition, $x + y + z = 1$ for each gun.

The next step is to find the u' and v' values for each color. These values correspond to the color's location on the CIE-UCS chromaticity diagram shown in Figure 2. The transformations are given by:

$$u' = 4x/(-2x + 12y + 3)$$

$$v' = 9y/(-2x + 12y + 3)$$

where

$$x = X_T/(X_T + Y_T + Z_T)$$
$$y = Y_T/(X_T + Y_T + Z_T)$$

The final step is to convert from (Y_T, u', v') to (L^*, u^*, v^*) coordinates. This is accomplished by the following set of equations:

$$L^* = 116\ (Y_T/Y_0)^{1/3} - 16$$

$$u^* = 13\ L^*\ (u' - u_0')$$

$$v^* = 13\ L^*\ (v' - v_0')$$

where

Y_0 = the value of Y_T when all three guns are turned on full blast, producing a pure white color

u_0' = the value of u' for pure white

v_0' = the value of v' for pure white

That's all. Pretty simple isn't it? The reverse transformations are equally as simple and are contained in Appendix B (which is just a listing of the program) among the coding of subroutine "RGBOUT."

Appendix B. FORTRAN Code and Documentation

Appendix B. FORTRAN Code and Documentation

```
C*****************************************************************************
C
C                              THESIS PROGRAM
C                                    BY
C                                ROSS ROLEY
C                                 GOR-84D
C
C*****************************************************************************
C
C                               MAIN PROGRAM
C
C          THE PURPOSE OF THIS PROGRAM IS TO MAXIMIZE THE MINIMUM
C     DISTANCE AMONG N POINTS IN A CONVEX THREE-DIMENSIONAL REGION
C     AND THEN SUCCESSIVELY MAXIMIZE THE SECOND SMALLEST DISTANCE,
C     THE THIRD SMALLEST, AND SO ON, UNTIL ALL POINTS ARE OPTIMALLY
C     SPACED.  THE REQUIRED INPUTS FOR THIS PROGRAM ARE THE NUMBER
C     OF POINTS BEING SPACED (N), THE DESIRED INITIAL STEP-SIZE (SIZE),
C     THE NUMBER OF CONSTRAINTS (FACES) ON THE REGION (F), AND THE
C     EQUATIONS OF THOSE CONSTRAINTS IN THE FORM  AX + BY + CZ ≤ K .
C     THE PROGRAM THEN OUTPUTS THE OPTIMAL VALUES OF X, Y, AND Z,
C     AND THE FIRST TEN (OR SOME OTHER NUMBER CHOSEN BY THE USER)
C     MINIMUM DISTANCES.
C
C     NOTE:  THROUGHOUT THIS PROGRAM, X, Y, AND Z ARE USED TO DENOTE
C     THE LOCATIONS OF THE POINTS BEING SPACED AND CORRESPOND TO
C     L*, U*, AND V* FOR THE COLOR SPACING PROBLEM.  THIS SHOULD NOT
C     BE CONFUSED WITH THE TRISTIMULUS VALUES OF A COLOR WHICH ARE
C     COMMONLY REFERRED TO AS CAPITAL X, Y, AND Z IN THE TRADE.  IN
C     THIS PROGRAM, THE TRISTIMULUS VALUES WILL BE REFERRED TO AS
C     XTOTAL, YTOTAL, AND ZTOTAL.
C
C
C          THERE ARE SEVERAL NOTES TO THE USER IN THE PROGRAM WHICH
C     ARE USED TO SIGNIFY PLACES WHERE YOU MAY BE REQUIRED TO MAKE
C     CHANGES.  THEY ARE SUMMARIZED HERE FOR YOUR CONVENIENCE.
C
C          IN THE MAIN PROGRAM, THE USER IS REQUIRED TO SUPPLY THE
C     NUMBER OF POINTS TO BE SPACED, THE SEED, THE MAXIMUM STEP-SIZE,
C     THE MINIMUM STEP-SIZE, AND THE STEP-SIZE REDUCING SCHEME.  IN
C     SUBROUTINE PLACE-POINTS, HE MUST SUPPLY THE MAXIMUM AND MIN-
C     IMUM VALUES ALLOWABLE FOR EACH OF THE THREE COORDINATES.  IN
C     SUBROUTINE FIND-DISTANCE, THE USER MUST CHOOSE THE NUMBER OF
C     DISTANCES HE WISHES TO APPEAR ON THE PROGRAM'S OUTPUT.
C
C          IN SUBROUTINE CHECK-POINT, THE USER IS REQUIRED TO SUPPLY
```

```
C       [illegible] AND THE EQUATIONS
C       FOR THOSE FACES IN THE FORM  AX + BY + CZ < K.  FOR THE
C       COLOR SPACING PROBLEM, THE NUMBER OF FACES SHOULD USUALLY
C       BE SEVEN.  IN ADDITION, ALL OTHER VARIABLES STAY THE SAME
C       EXCEPT FOR THE VALUES OF K;  K(1) IS THE MAXIMUM ALLOWABLE
C       LUMINANCE YOU WISH TO SET FOR THE RED GUN, K(2) = THE MAXIMUM
C       GREEN GUN LUMINANCE, K(3) = THE MAXIMUM BLUE GUN LUMINANCE,
C       AND K(7) = THE MINIMUM TOTAL LUMINANCE YOU WISH TO PERMIT (IT
C       SHOULD BE ENTERED AS A NEGATIVE VALUE).
C
C            IN SUBROUTINE [illegible] THE ONLY REQUIRED CHANGE IS THE
C       VALUE OF THE MAXIMUM STEP-SIZE.
C
C            FOR COLOR SPACING PROBLEMS, SUBROUTINE RGBOUT IS VERY
C       IMPORTANT.  THERE, THE USER MUST SUPPLY THE CHROMATICITY
C       COORDINATES OF THE GUNS, THE MAXIMIM LUMINANCE ALLOWABLE
C       (L-NOT), AND THE VALUES OF U PRIME NOT AND V-PRIME NOT.  THAT
C       IS THE EXTENT OF THE CHANGES THAT NEED TO BE MADE FOR ANY
C       NORMAL USE OF THIS PROGRAM.
C
C
C*********************************************************************
C
C                        VARIABLE DEFINITIONS
C
C       REAL VARIABLES:
C       X(I)= A ONE-DIMENSIONAL ARRAY CONTAINING THE X-COORDINATE OF
C             POINT I
C       Y(I)= THE Y-COORDINATE OF POINT I
C       Z(I)= THE Z-COORDINATE OF POINT I
C       D(I,J)= THE EUCLIDEAN DISTANCE BETWEEN POINT I AND POINT J
C       CAPD= THE CURRENT MINIMUM DISTANCE BETWEEN ALL PAIRS OF POINTS
C       NEWD= THE NEW MINIMUM DISTANCE
C       SIZE= THE STEP-SIZE
C       ENDX(I)= THE X-COORDINATE OF ENDPOINT I, WHERE THE ENDPOINTS
C                ARE THE TWO CLOSEST POINTS IN THE REGION
C       ENDY(I)= THE Y-COORDINATE OF ENDPOINT I
C       ENDZ(I)= THE Z-COORDINATE OF ENDPOINT I
C       ALTX(I,J)= THE JTH ALTERNATE LOCATION FOR THE X-COORDINATE OF
C                  ENDPOINT I
C       ALTY(I,J)= THE JTH ALTERNATE LOCATION FOR THE Y-COORDINATE OF
C                  ENDPOINT I
C       ALTZ(I,J)= THE JTH ALTERNATE LOCATION FOR THE Z-COORDINATE OF
C                  ENDPOINT I
C       MEMX(I)= THE X-COORDINATE OF POINT I FROM THE PREVIOUS ITERATION
C       MEMY(I)= THE Y-COORDINATE OF POINT I FROM THE PREVIOUS ITERATION
C       MEMZ(I)= THE Z-COORDINATE OF POINT I FROM THE PREVIOUS ITERATION
C       DUM1,DUM2,DUM3= DUMMY VARIABLES REQUIRED FOR SUBROUTINE RGBOUT
C                       BUT NOT USED IN THE MAIN PROGRAM
C       SEED= THE STARTING VALUE FOR THE RANDOM NUMBER GENERATING SEQUENCE
C             IN THE CYBER KNOWN AS SUBROUTINE 'RANSET'
C
```

```
C       [illegible]
C       N   THE NUMBER OF POINTS REQUIRING SPACING
C       ENDPT1  THE NUMBER ASSOCIATED WITH ENDPOINT 1
C       ENDPT2  THE NUMBER ASSOCIATED WITH ENDPOINT 2
C
C       CHARACTER VARIABLES:
C       STATUS  THE STATUS OF THE PROGRAM; 'RUN' IF THE STOPPING
C                 CRITERION HAS NOT BEEN MET, AND 'DONE' IF IT HAS
C       STATPT(I)  THE STATUS OF POINT I: 'F' FOR FIXED, AND 'U'
C                     FOR UNFIXED
C
C       EXTERNAL VARIABLES:
C       RANF   VARIABLE USED IN THE RANDOM NUMBER GENERATING SUBROUTINE
C              IN THE CYBER CALLED 'RANSET'
C
C
C*****************************************************************************
C
C
        PROGRAM SPACPT
        REAL  X(50),Y(50),Z(50),D(50,50),CAPD,NEWD,SIZE,ENDX(2),ENDY(2),
       &      ENDZ(2),ALTX(2,27),ALTY(2,27),ALTZ(2,27),MEMX(50),MEMY(50),
       &      MEMZ(50),DUM1(50),DUM2(50),DUM3(50),SEED
        INTEGER  N,ENDPT1,ENDPT2
        CHARACTER  STATUS*4,STATPT(50)*1
        EXTERNAL  RANF
        OPEN (3,FILE='SPCOUT')
        REWIND 3
        IO=3
        IOO=3
C
C                       -- IMPORTANT NOTE TO USER --
C
C       THE USER MUST SUPPLY THE VALUE OF N AND SIZE HERE, AND THE SIZE
C       AGAIN IN SUBROUTINE 'FIX-POINT.'  I RECOMMEND USING A STEP-SIZE
C       APPROXIMATELY EQUAL TO ONE-HALF THE DISTANCE BETWEEN THE TWO
C       FARTHEST POINTS IN THE REGION, BUT THE USER IS ENCOURAGED TO TRY
C       THE SAME PROBLEM WITH SEVERAL DIFFERENT MAXIMUM STEP-SIZES TO
C       INCREASE THE CHANCES OF GETTING AN OPTIMAL SOLUTION.
C
        SIZE=130.0
        N=10
        STATUS='DONE'
        DO 100 I=1,N
            STATPT(I)='U'
            MEMX(I)=0.0
            MEMY(I)=0.0
            MEMZ(I)=0.0
100     CONTINUE
C
C                       -- IMPORTANT NOTE TO USER --
C
```

```
C      THE SEED IS THE STARTING VALUE FOR THE RANDOM NUMBER GENERATOR
C      SEQUENCE WHICH RANDOMLY CHOOSES THE INITIAL LOCATIONS FOR THE
C      POINTS.  IT IS RECOMMENDED THAT MULTIPLE RUNS BE PERFORMED FOR
C      ALL PROBLEMS, IN WHICH CASE THE USER SHOULD CHANGE THE SEED FOR
C      EACH RUN.  IT CAN BE ANY NINE DIGIT INTEGER.
C
       SEED=789546121.0
       WRITE(3,110)SEED
110    FORMAT(16X,'THE SEED = ',F11.1)
C  THIS SECTION CALCULATES THE INITIAL LOCATIONS FOR THE POINTS AND
C  FORMATS AND PRINTS THE LOCATIONS.
       CALL RANSET(SEED)
       CALL PLACPT(N,X,Y,Z,SEED)
       WRITE(3,120)
120    FORMAT(11X,'THESE ARE THE STARTING LOCATIONS')
       CALL PRINT(X,Y,Z,N)
       CALL RGBOUT(X,Y,Z,N,DUM1,DUM2,DUM3,STATUS)
       STATUS='RUN'
C  THE MEAT OF THE PROGRAM BEGINS HERE
130    CALL FINDD(X,Y,Z,N,STATPT,CAPD,ENDX,ENDY,ENDZ,ENDPT1,ENDPT2,
      & STATUS)
       CALL DEFALT(ENDX,ENDY,ENDZ,SIZE,ENDPT1,ENDPT2,STATPT,
      & ALTX,ALTY,ALTZ)
       CALL CHKALT(X,Y,Z,N,CAPD,ALTX,ALTY,ALTZ,STATPT,ENDPT1,ENDPT2,NEWD)
C  IF THE MINIMUM DISTANCE IS GREATER, THEN KEEP ITERATING
       IF (NEWD .GT. CAPD) THEN
            CAPD=NEWD
            GO TO 130
C  IF NOT, THEN REDUCE THE STEP-SIZE
       ELSE
            SIZE=SIZE/2.0
            IF (SIZE .LE. .0001) THEN
C  NOTE:   THE USER MAY WISH TO USE A DIFFERENT MINIMUM STEP-SIZE.
C          FOR THREE DECIMAL PLACES OF ACCURACY, I USE .0001.  YOU
C          MAY ALSO WISH TO USE A SCHEME OTHER THAN HALVING THE STEP-
C          SIZE, LIKE REDUCING IT BY THREE-FOURTHS INSTEAD.
                 CALL FIXPT(X,Y,Z,N,STATPT,ENDPT1,ENDPT2,SIZE,STATUS,
      &           MEMX,MEMY,MEMZ)
                 IF (STATUS .EQ. 'RUN') THEN
                      GO TO 130
                 ELSE IF (STATUS .EQ. 'DONE') THEN
                      GO TO 140
                 END IF
            ELSE
                 GO TO 130
            END IF
       END IF
C  THIS SECTION CONTROLS THE FORMAT OF THE RESULTS
140    WRITE(3,150)
150    FORMAT(15X,'THIS IS THE FINAL ANSWER')
       CALL PRINT(X,Y,Z,N)
       CALL RGBOUT(X,Y,Z,N,DUM1,DUM2,DUM3,STATUS)
```

```
     [illegible]
    & STATUS)
      END
C
C
C*****************************************************************************
C
C                            SUBROUTINE PLACE-POINTS
C
C           THE PURPOSE OF THIS BABY IS TO RANDOMLY PLACE THE POINTS
C      IN THE REGION.  THE LITTLE TYKE RECEIVES THE NUMBER OF POINTS
C      (N) AS AN INPUT, AND GURGLES UP THE STARTING LOCATIONS FOR THE
C      POINTS (X, Y, AND Z) AS OUTPUTS.
C
C      NOTE:  THIS SUBROUTINE USES THE RANDOM NUMBER GENERATING FUNCTION
C      IN THE CYBER CALLED 'RANSET.'  MOST MAIN-FRAME COMPUTERS HAVE
C      A RANDOM NUMBER GENERATING CAPABILITY, SO CONSULT YOUR LOCAL
C      COMPUTER EXPERT FOR DIRECTIONS ON HOW TO USE IT ON YOUR FAVORITE
C      SYSTEM.
C
C
C*****************************************************************************
C
C      .                       VARIABLE DEFINITIONS
C
C      REAL VARIABLES:
C      X(I)= THE INITIAL X-COORDINATE OF POINT I
C      Y(I)= THE INITIAL Y-COORDINATE OF POINT I
C      Z(I)= THE INITIAL Z-COORDINATE OF POINT I
C      YRED(I)= THE RED PHOSPHOR LUMINANCE FOR POINT I
C      YGREEN(I)= THE GREEN PHOSPHOR LUMINANCE FOR POINT I
C      YBLUE(I)= THE BLUE PHOSPHOR LUMINANCE FOR POINT I
C      SEED= THE STARTING VALUE FOR THE RANDOM NUMBER GENERATING SEQUENCE
C            IN THE CYBER KNOWN AS SUBROUTINE 'RANSET'
C
C      INTEGER VARIABLES:
C      N= THE NUMBER OF POINTS BEING LOCATED
C      J= THE NUMBER OF POINTS BEING TRANSFORMED BY SUBROUTINE 'RGBOUT'
C
C      CHARACTER VARIABLES:
C      POINT= THE LOCATION OF THE POINT; EITHER 'IN' OR 'OUT' OF THE
C             REGION
C      DUM1= A DUMMY VARIABLE REQUIRED FOR SUBROUTINE 'RGBOUT' BUT NOT
C            USED IN THIS SUBROUTINE
C
C      EXTERNAL VARIABLES:
C      RANF= A VARIABLE USED IN CONJUNCTION WITH 'RANSET' TO GENERATE
C            A RANDOM NUMBER BETWEEN ZERO AND ONE
C
C
C*****************************************************************************
C
```

```
C
      SUBROUTINE PLACPT(N,X,Y,Z,SEED)
      REAL   X(50),Y(50),Z(50),SEED,YRED(50),YGREEN(50),YBLUE(50)
      INTEGER   N,J
      CHARACTER   POINT*3,DUM1*4
      EXTERNAL   RANF
      DO 160 I=1,N
170         X(I)=RANF()*(100.0-9.0)+9.0
            Y(I)=RANF()*(175.0+75.0)-75.0
            Z(I)=RANF()*(100.0+125.0)-125.0
C
C                      -- IMPORTANT NOTE TO USER --
C
C     THE VARIABLE 'RANF' GIVES VALUES IN THE RANGE FROM ZERO TO ONE.
C     IF THE USER WANTS VALUES OUTSIDE OF THAT RANGE, THE EQUATIONS
C     FOR X, Y, AND Z, SHOULD LOOK LIKE THIS:
C                      X(I)=RANF()*(XMAX-XMIN)+XMIN
C                      Y(I)=RANF()*(YMAX-YMIN)+YMIN
C                      Z(I)=RANF()*(ZMAX-ZMIN)+ZMIN
C     WHERE MAX AND MIN ARE THE MAXIMUM AND MINIMUM POSSIBLE VALUES
C     FOR X, Y, AND Z, RESPECTIVELY.
C
            J=1
            CALL RGBOUT(X(I),Y(I),Z(I),J,YRED(I),YGREEN(I),YBLUE(I),DUM1)
            CALL CHEKPT(YRED(I),YGREEN(I),YBLUE(I),POINT)
            IF (POINT .EQ. 'OUT') THEN
                  GO TO 170
            END IF
160   CONTINUE
      END
C
C
C*********************************************************************
C
C                       SUBROUTINE FIND-DISTANCE
C
C           THE PURPOSE OF THIS SUBROUTINE IS TO FIND THE MINIMUM DIS-
C     TANCE BETWEEN ALL N(N-1)/2 PAIRS OF (UNFIXED) POINTS.  THE
C     INPUTS FOR THIS PROGRAM ARE THE LOCATIONS OF THE POINTS (X,
C     Y, AND Z), THE STATUS OF THOSE POINTS AS TO WHETHER THEY ARE
C     FIXED OR UNFIXED (STATPT), AND THE NUMBER OF POINTS (N).  THE
C     PROGRAM THEN OUTPUTS THE MINIMUM EUCLIDEAN DISTANCE BETWEEN
C     ALL PAIRS OF (UNFIXED) POINTS, THE TWO POINTS THAT CORRESPOND
C     TO THAT MINIMUM DISTANCE (ENDPT1 AND ENDPT2), AND THE
C     COORDINATES OF THOSE POINTS (ENDX, ENDY, AND ENDZ).
C
C
C*********************************************************************
C
C                         VARIABLE DEFINITIONS
C
C     REAL VARIABLES:
```

```
C      [illegible]
C      Y(I)= THE Y-COORDINATE OF POINT I
C      Z(I)= THE Z-COORDINATE OF POINT I
C      CAPD= THE MINIMUM EUCLIDEAN DISTANCE BETWEEN ALL (UNFIXED) PAIRS
C             OF POINTS
C      ENDX(I)= THE X-COORDINATE OF ENDPOINT I
C      ENDY(I)= THE Y-COORDINATE OF ENDPOINT I
C      ENDZ(I)= THE Z-COORDINATE OF ENDPOINT I
C      D(I,J)= THE EUCLIDEAN DISTANCE FROM POINT I TO POINT J
C      DIS(K)= THE KTH SMALLEST DISTANCE
C
C      INTEGER VARIABLES:
C      N= THE NUMBER OF POINTS
C      ENDPT1= THE NUMBER ASSOCIATED WITH ENDPOINT 1
C      ENDPT2= THE NUMBER ASSOCIATED WITH ENDPOINT 2
C      K= THE COUNTER USED IN ORDERING THE DISTANCES
C      L= THE NUMBER OF DISTANCES THE USER WISHES TO PRINT
C      EYE(K)= AN ENDPOINT COORESPONDING TO THE KTH SMALLEST D(I,J)
C      JAY(K)= THE OTHER ENDPOINT COORESPONDING TO THE KTH SMALLEST D(I,J)
C
C      CHARACTER VARIABLES:
C      STATPT(I)= THE STATUS OF POINT I; EITHER FIXED OR UNFIXED
C      STATUS= THE STATUS OF THE PROGRAM; EITHER 'DONE' OR 'RUN'
C
C
C***********************************************************************
C
C
      SUBROUTINE FINDD(X,Y,Z,N,STATPT,CAPD,ENDX,ENDY,ENDZ,ENDPT1,ENDPT2,
     & STATUS)
      REAL  X(50),Y(50),Z(50),CAPD,ENDX(2),ENDY(2),ENDZ(2),D(50,50),
     &       DIS(300)
      INTEGER  N,ENDPT1,ENDPT2,K,L,EYE(300),JAY(300)
      CHARACTER  STATPT(50)*1,STATUS*4
      CAPD=10000.0
      DO 200 I=1,N-1
          DO 210 J=I+1,N
C  UNLESS BOTH POINTS ARE FIXED, CALCULATE THE DISTANCE BETWEEN THEM
                IF ((STATPT(I) .EQ. 'U') .OR. (STATPT(J) .EQ. 'U'))
     &           THEN
C  THIS IS THE EUCLIDEAN DISTANCE FORMULA
                     D(I,J)=SQRT((X(I)-X(J))**2+(Y(I)-Y(J))**2+
     %                (Z(I)-Z(J))**2)
                     IF (D(I,J) .LE. CAPD) THEN
                         CAPD=D(I,J)
                         ENDPT1=I
                         ENDPT2=J
                     END IF
                END IF
210       CONTINUE
200   CONTINUE
      ENDX(1)=X(ENDPT1)
      ENDY(1)=Y(ENDPT1)
      ENDZ(1)=Z(ENDPT1)
      ENDX(2)=X(ENDPT2)
      ENDY(2)=Y(ENDPT2)
      ENDZ(2)=Z(ENDPT2)
C   THIS SECTION ORDERS THE DISTANCES AND PRINTS THE L SMALLEST
C   DISTANCES TO OUTPUT WHEN THE PROGRAM IS DONE
      IF (STATUS .EQ. 'DONE') THEN
      L=10
C   THE USER CONTROLS THE NUMBER OF DISTANCES HE WISHES TO SEE BY
C   CHANGING THE VALUE OF L HERE.  HE CAN CHOOSE TO SEE UP TO 300
C   DISTANCES.
            DO 220 K=1,300
                  DIS(K)=1000.0
220         CONTINUE
            DO 230 I=1,N-1
                  DO 240 J=I+1,N
                        DO 250 K=1,L
                              IF (D(I,J) .LT. DIS(K)) THEN
                                    DO 260 M=L,K+1,-1
                                          DIS(M)=DIS(M-1)
                                          EYE(M)=EYE(M-1)
                                          JAY(M)=JAY(M-1)
260                                 CONTINUE
                                    DIS(K)=D(I,J)
                                    EYE(K)=I
                                    JAY(K)=J
                                    GO TO 240
                              END IF
250                     CONTINUE
240               CONTINUE
230         CONTINUE
            WRITE(3,280)L
            DO 290 K=1,L
                  WRITE(3,295)EYE(K),JAY(K),DIS(K)
290         CONTINUE
            WRITE(3,270)
270         FORMAT(' ')
280         FORMAT(9X,'THESE ARE THE ',I3,' SMALLEST DISTANCES')
295         FORMAT(19X,'D(',I2,',',I2,')=',F7.3)
      END IF
      END
C
C
C*************************************************************************
C
C                       SUBROUTINE DEFINE-ALTERNATIVES
C
C           THE PURPOSE OF THIS SUBROUTINE IS TO DEFINE THE VARIOUS
C     ALTERNATIVE LOCATIONS FOR ENDPOINTS 1 AND 2.  THE PROGRAM
C     IS PASSED THE COORDINATES OF THE ENDPOINTS AND THE STEP-SIZE
C     AND OUTPUTS A TWO BY TWENTY-SEVEN ARRAY CONTAINING THE
```

```
C       COORDINATES OF THE ALTERNATIVES.  THESE ALTERNATIVES LIE
C       ON A CUBE WHOSE SIDES ARE TWICE THE STEP-SIZE IN DISTANCE
C       WITH THE ENDPOINT AT ITS CENTER.
C
C
C*********************************************************************
C
C                                 VARIABLE DEFINITIONS
C
C       REAL VARIABLES:
C       ENDX(I)= THE X-COORDINATE OF ENDPOINT I
C       ENDY(I)= THE Y-COORDINATE OF ENDPOINT I
C       ENDZ(I)= THE Z-COORDINATE OF ENDPOINT I
C       SIZE= THE STEP-SIZE
C       ALTX(I,J)= THE X-COORDINATE OF THE JTH ALTERNATIVE FOR ENDPOINT I
C       ALTY(I,J)= THE Y-COORDINATE OF THE JTH ALTERNATIVE FOR ENDPOINT I
C       ALTZ(I,J)= THE Z-COORDINATE OF THE JTH ALTERNATIVE FOR ENDPOINT I
C
C       INTEGER VARIABLES:
C       ENDPT1= THE NUMBER CORRESPONDING TO ENDPOINT 1
C       ENDPT2= THE NUMBER CORRESPONDING TO ENDPOINT 2
C       P= THE INDEX FOR THE ALTERNATE LOCATIONS
C
C       CHARACTER VARIABLES:
C       STATPT(I)= THE STATUS OF POINT I; EITHER FIXED OR UNFIXED
C
C
C*********************************************************************
C
C
        SUBROUTINE DEFALT(ENDX,ENDY,ENDZ,SIZE,ENDPT1,ENDPT2,STATPT,ALTX,
      %                      ALTY,ALTZ)
        REAL   ENDX(2),ENDY(2),ENDZ(2),SIZE,ALTX(2,27),ALTY(2,27),
      %        ALTZ(2,27)
        INTEGER   ENDPT1,ENDPT2,P
        CHARACTER   STATPT(50)*1
C   THESE LOOPS LOAD THE ALTERNATE LOCATIONS FOR ENDPOINTS 1 AND 2
        DO 300 T=1,2
              P=0
              DO 310 S=-1,1
                    DO 320 R=-1,1
                          DO 330 Q=-1,1
                                P=P+1
                                ALTX(T,P)=ENDX(T)-S*SIZE
                                ALTY(T,P)=ENDY(T)-R*SIZE
                                ALTZ(T,P)=ENDZ(T)-Q*SIZE
330                       CONTINUE
320                 CONTINUE
310           CONTINUE
300     CONTINUE
C   IF ONE OF THE ENDPOINTS IS FIXED, THEN IT HAS NO ALTERNATE POSITIONS
        IF (STATPT(ENDPT1) .EQ. 'F') THEN
```

```
            DO 340 S=1,27
                  ALTX(1,S)=ENDX(1)
                  ALTY(1,S)=ENDY(1)
                  ALTZ(1,S)=ENDZ(1)
340         CONTINUE
        END IF
        IF (STATPT(ENDPT2) .EQ. 'F') THEN
            DO 350 S=1,27
                  ALTX(2,S)=ENDX(2)
                  ALTY(2,S)=ENDY(2)
                  ALTZ(2,S)=ENDZ(2)
 350        CONTINUE
        END IF
        END
C
C
C**********************************************************************
C
C                         SUBROUTINE CHECK-ALTERNATIVES
C
C             THIS SUBROUTINE CHECKS THE VARIOUS ALTERNATIVES TO SEE
C       IF THEY LIE IN THE REGION, THEN TESTS THE REMAINING CANDIDATES
C       TO SEE IF THEY INCREASE OR DECREASE THE DISTANCE BETWEEN THE
C       ENDPOINTS.  IF THE DISTANCE IS INCREASED, THIS PROGRAM CALLS
C       SUBROUTINE 'FIND-DISTANCE' TO SEE IF THE OVERALL MINIMUM
C       DISTANCE IS INCREASED OR DECREASED.  IF SO, THE ENDPOINTS
C       ARE MOVED TO THE ALTERNATE LOCATION THAT MOST INCREASES THE
C       OVERALL MINIMUM DISTANCE WITH TIES GOING TO THE ALTERNATIVE
C       THAT INCREASES THE ENDPOINT DISTANCE THE MOST.  IT THEN PASSES
C       THOSE NEW LOCATIONS TO THE MAIN PROGRAM.
C
C
C**********************************************************************
C
C                             VARIABLE DEFINITIONS
C
C       REAL VARIABLES:
C       X(I)= THE X-COORDINATE OF POINT I
C       Y(I)= THE Y-COORDINATE OF POINT I
C       Z(I)= THE Z-COORDINATE OF POINT I
C       CAPD= THE MINIMUM DISTANCE BETWEEN ALL PAIRS OF (UNFIXED) POINTS
C       ALTX(I,J)= THE X-COORDINATE OF THE JTH ALTERNATIVE FOR ENDPOINT I
C       ALTY(I,J)= THE Y-COORDINATE OF THE JTH ALTERNATIVE FOR ENDPOINT I
C       ALTZ(I,J)= THE Z-COORDINATE OF THE JTH ALTERNATIVE FOR ENDPOINT I
C       NEWD= THE NEW MINIMUM DISTANCE BETWEEN ALL THE POINTS
C       CHKX(I)= THE X-COORDINATE OF POINT I BEING CHECKED AS A NEW
C                LOCATION SCHEME BY SUBROUTINE 'FIND-DISTANCE'
C       CHKY(I)= THE Y-COORDINATE OF POINT I BEING CHECKED
C       CHKZ(I)= THE Z-COORDINATE OF POINT I BEING CHECKED
C       DUM1,DUM2,DUM3= DUMMY VARIABLES THAT ARE RETURNED BY SUBROUTINE
C                       'FIND-DISTANCE' BUT ARE NOT USED IN THIS SUBROUTINE
C       MIND= THE MINIMUM DISTANCE CALCULATED BY SUBROUTINE 'FIND-DISTANCE'
```

```
C                FOR THE LOCATION SCHEME DEFINED BY CHKX,CHKY, AND CHKZ
C         MAXEND= THE VARIABLE USED AS A TIE-BREAKER FOR ALTERNATIVES WITH
C                 IDENTICAL MINIMUM DISTANCES
C         ENDPTD= THE DISTANCE BETWEEN ENDPOINTS 1 AND 2
C         YRED(I,J)= THE VALUE OF ALTX(I,J) IN RED-GREEN-BLUE SPACE
C         YGREEN(I,J)= THE VALUE OF ALTY(I,J) IN RGB SPACE
C         YBLUE(I,J)= THE VALUE OF ALTZ(I,J) IN RGB SPACE
C
C         INTEGER VARIABLES:
C         ENDPT1= THE NUMBER ASSOCIATED WITH ENDPOINT 1
C         ENDPT2= THE NUMBER ASSOCIATED WITH ENDPOINT 2
C         N= THE NUMBER OF POINTS
C         DUM4,DUM5= NOT USED IN THIS SUBROUTINE
C
C         CHARACTER VARIABLES:
C         STATPT(I)= THE STATUS OF POINT I; EITHER FIXED OR UNFIXED
C         LOCALT(I,J)= THE LOCATION OF THE JTH ALTERNATIVE FOR ENDPOINT I;
C                         EITHER IN OR OUT
C         DUM6= A VARIABLE REQUIRED FOR SUBROUTINE 'RGBOUT' BUT NOT USED
C               IN THIS SUBROUTINE
C
C
C************************************************************************
C
C
          SUBROUTINE CHKALT(X,Y,Z,N,CAPD,ALTX,ALTY,ALTZ,STATPT,ENDPT1,
         &                     ENDPT2,NEWD)
          REAL  X(50),Y(50),Z(50),CAPD,ALTX(2,27),ALTY(2,27),ALTZ(2,27),
         &      NEWD,CHKX(50),CHKY(50),CHKZ(50),DUM1(2),DUM2(2),DUM3(2),
         &      MIND,MAXEND,ENDPTD,YRED(2,27),YGREEN(2,27),YBLUE(2,27)
          INTEGER  ENDPT1,ENDPT2,N,DUM4,DUM5
          CHARACTER  STATPT(50)*1,LOCALT(2,27)*3,DUM6*4
          NEWD=CAPD
          MAXEND=CAPD
          DO 400 I=1,N
                CHKX(I)=X(I)
                CHKY(I)=Y(I)
                CHKZ(I)=Z(I)
400       CONTINUE
C     THIS LOOP CHECKS ALL 54 ALTERNATIVES TO SEE IF THEY ARE IN OR OUT
C     OF THE REGION
          DO 410 S=1,2
                DO 420 T=1,27
                      CALL RGBOUT(ALTX(S,T),ALTY(S,T),ALTZ(S,T),1,YRED(S,T),
         &             YGREEN(S,T),YBLUE(S,T),DUM6)
                      CALL CHEKPT(YRED(S,T),YGREEN(S,T),YBLUE(S,T),
         &             LOCALT(S,T))
420             CONTINUE
410       CONTINUE
          DO 430 T=1,27
                IF (LOCALT(1,T) .EQ. 'IN') THEN
                      DO 440 S=1,27
```

```
                          IF (T .NE. S .AND. LOCAL(2,S) .EQ. 'IN') THEN
C   IF BOTH ENDPOINTS ARE STILL IN, THEN CALCULATE THE NEW DISTANCE
C   BETWEEN THEM
                            ENDPTD=SQRT((ALTX(1,T)-ALTX(2,S))**2+
     %                       (ALTY(1,T)-ALTY(2,S))**2+(ALTZ(1,T)-
     %                      ALTZ(2,S))**2)
C   IF THAT DISTANCE IS GREATER, THEN CHECK THE OVERALL MINIMUM DISTANCE
                            IF (ENDPTD .GT. NEWD) THEN
                               CHKX(ENDPT1)=ALTX(1,T)
                               CHKY(ENDPT1)=ALTY(1,T)
                               CHKZ(ENDPT1)=ALTZ(1,T)
                               CHKX(ENDPT2)=ALTX(2,S)
                               CHKY(ENDPT2)=ALTY(2,S)
                               CHKZ(ENDPT2)=ALTZ(2,S)
                               CALL FINDD(CHKX,CHKY,CHKZ,N,STATPT,MIND,
     %                          DUM1,DUM2,DUM3,DUM4,DUM5,DUM6)
C   IF THE MINIMUM DISTANCE IS THE SAME BUT WITH A GREATER ENDPOINT
C   DISTANCE, OR IF THE MINIMUM DISTANCE IS BETTER, THEN MOVE THE
C   ENDPOINTS
                               IF ((MIND .GT. NEWD) .OR. (MIND .EQ.
     %                          NEWD .AND. ENDPTD .GT. MAXEND)) THEN
                                  NEWD=MIND
                                  MAXEND=ENDPTD
                                  X(ENDPT1)=ALTX(1,T)
                                  Y(ENDPT1)=ALTY(1,T)
                                  Z(ENDPT1)=ALTZ(1,T)
                                  X(ENDPT2)=ALTX(2,S)
                                  Y(ENDPT2)=ALTY(2,S)
                                  Z(ENDPT2)=ALTZ(2,S)
                               END IF
                            END IF
                         END IF
440                   CONTINUE
                  END IF
430       CONTINUE
          END
C
C
C*************************************************************************
C
C                         SUBROUTINE CHECK-POINT
C
C              THIS SUBROUTINE CHECKS POINTS TO SEE IF THEY ARE IN OR
C         OUT OF THE REGION.   IT CHECKS THEM AGAINST A SET OF EQUATIONS
C         (CONSTRAINTS) OF THE FORM AX + BY + CZ ≤ K WHICH MUST BE
C         SUPPLIED BY THE USER, AND THEN OUTPUTS A CHARACTER VARIABLE
C         WHICH IDENTIFIES WHETHER THE POINT IS 'IN' OR 'OUT.'
C
C         NOTE:  THIS SUBROUTINE IS FOR LINEAR CONSTRAINTS ONLY.  NON-
C         LINEAR CONSTRAINTS MAY BE USED BY CHANGING THE EQUATION FOR
C         V(S), BUT THEY MUST BE CONVEX IN ORDER FOR THE ALGORITHM TO
C         WORK.
```

```
C
C
C
C*********************************************************************
C
C                            VARIABLE DEFINITIONS
C
C      REAL VARIABLES:
C      XCOORD  THE X-COORDINATE OF THE POINT BEING TESTED
C      YCOORD  THE Y-COORDINATE OF THE POINT BEING TESTED
C      ZCOORD  THE Z-COORDINATE OF THE POINT BEING TESTED
C      A(I)= THE COEFFICIENT OF X IN CONSTRAINT I
C      B(I)= THE COEFFICIENT OF Y IN CONSTRAINT I
C      C(I)= THE COEFFICIENT OF Z IN CONSTRAINT I
C      K(I)= THE VALUE OF THE RIGHT HAND SIDE IN CONSTRAINT I
C      V(I)= THE VALUE OF CONSTRAINT I EVALUATED AT THE POINT
C            BEING TESTED
C
C      INTEGER VARIABLES:
C      F= THE NUMBER OF FACES (CONSTRAINTS) ON THE REGION
C
C      CHARACTER VARIABLES:
C      POINT= THE LOCATION OF THE POINT; EITHER IN OR OUT
C
C
C*********************************************************************
C
C
      SUBROUTINE CHEKPT(XCOORD,YCOORD,ZCOORD,POINT)
      REAL  XCOORD,YCOORD,ZCOORD,A(10),B(10),C(10),K(10),V(10)
      INTEGER  F
      CHARACTER  POINT*3
C
C                      -- IMPORTANT NOTE TO USER --
C
C      THE USER MUST SUPPLY THE NUMBER OF FACES FOR EACH NEW PROBLEM HERE.
C
      F=7
C  THIS LOOP INITIALIZES THE COEFFICIENTS AT ZERO
      DO 500 S=1,F
           A(S)=0.0
           B(S)=0.0
           C(S)=0.0
           K(S)=0.0
           V(S)=0.0
500   CONTINUE
C
C                      -- IMPORTANT NOTE TO USER --
C
C      THE USER MUST SUPPLY THE EQUATIONS FOR THE CONSTRAINTS IN THE
C      FORM AX + BY + CZ ≤ K  I.E. HE MUST DETERMINE AND PROVIDE
C      THE VALUES OF A(S), B(S), C(S), AND K(S) FOR S= 1 TO F.  FOR
```

```
C       INSTANCE, IF THE FIRST CONSTRAINT IS 2X + 3Y + 4Z _ 5, THEN
C       A(1)=2, B(1)=3, C(1)=4, AND K(1)=5.  IF THE SECOND CONSTRAINT
C       IS X ≤ 1, THEN A(2)=1, B(2)=0, C(2)=0, AND K(2)=1.  IF A
C       CONSTRAINT HAS A GREATER THAN SIGN, THEN REVERSE THE SIGN BY
C       NEGATING THE COEFFICIENTS.  FOR INSTANCE, X + Y + Z ≥ 3 BECOMES
C       -X -Y -Z ≤ -3, SO A(3)=-1, B(3)=-1, C(3)=-1, AND K(3)=-1.  IF
C       THE CONSTRAINT IS AN EQUALITY CONSTRAINT, THEN FORM TWO EQUATIONS
C       WITH INEQUALITIES.  FOR INSTANCE, Y + Z = 5 BECOMES Y + Z ≤ 5
C       AND -Y -Z ≥ -5, SO A(4)=0, B(4)=1, C(4)=1, K(4)=5, A(5)=0,
C       B(5)=-1, C(5)=-1, AND K(5)=-5.  IF YOU HAVE ANY PROBLEMS,
C       CONSULT YOUR LOCAL MATH EXPERT.
C
        A(1)=1.0
        B(2)=1.0
        C(3)=1.0
        A(4)=-1.0
        B(5)=-1.0
        C(6)=-1.0
        A(7)=-1.0
        B(7)=-1.0
        C(7)=-1.0
        K(1)=50.0
        K(2)=160.0
        K(3)=20.0
        K(4)=0.0
        K(5)=0.0
        K(6)=0.0
        K(7)=-2.3
C   THE CONSTRAINTS AS THEY ARE SHOWN ABOVE CORRESPOND TO CARTER AND
C   CARTER'S CONCEPT OF THE FEASIBLE REGION FOR THE COLOR SPACING
C   PROBLEM AND LOOK LIKE THIS IN RED, GREEN, AND BLUE COORDINATES:
C                          RED ≤ 50
C                          GREEN ≤ 160
C                          BLUE ≤ 20
C                          RED ≥ 0
C                          GREEN ≥ 0
C                          BLUE ≥ 0
C                          RED + GREEN + BLUE ≥ 2.3
        POINT='IN'
C   THIS LOOP CHECKS THE POINT AGAINST EACH OF THE CONSTRAINTS
        DO 510 S=1,F
             V(S)=A(S)*XCOORD+B(S)*YCOORD+C(S)*ZCOORD
             IF (V(S) .GT. K(S)) THEN
                  POINT='OUT'
                  GO TO 520
             END IF
510     CONTINUE
520     END
C
C
C**********************************************************************
C
```

```
C                             SUBROUTINE FIX POINT
C
C            THE PURPOSE OF THIS SUBROUTINE IS TO FIX POINTS AND
C      DETERMINE WHETHER OR NOT THE PROGRAM IS DONE.  THE LOGIC
C      ASSOCIATED WITH FIXING POINTS IS THAT, ONCE THE MINIMUM
C      DISTANCE IS MAXIMIZED, THAT DISTANCE IS NO LONGER UNDER CONSIDER-
C      ATION, SO ONE OF THE ENDPOINTS IS FIXED AT ITS LOCATION AND THE
C      SECOND CLOSEST PAIR OF POINTS IS MAXIMIZED.  THIS CONTINUES
C      UNTIL ALL POINTS ARE FIXED, AT WHICH TIME THEY ARE UNFIXED
C      TO SEE IF ANY IMPROVEMENTS CAN BE MADE.  THE PROGRAM STOPS
C      WHEN NONE CAN BE MADE.
C
C
C*******************************************************************
C
C                          VARIABLE DEFINITIONS
C
C      REAL VARIABLES:
C      X(I)= THE X-COORDINATE OF POINT I
C      Y(I)= THE Y-COORDINATE OF POINT I
C      Z(I)= THE Z-COORDINATE OF POINT I
C      DUM1,DUM2,DUM3,DUM4= DUMMY VARIABLES REQUIRED FOR SUBROUTINE 'FIND-
C                           DISTANCE' BUT NOT USED IN THIS SUBROUTINE
C      MEMX(I)= THE X-COORDINATE OF POINT I FROM THE PREVIOUS ITERATION
C      MEMY(I)= THE Y-COORDINATE OF POINT I FROM THE PREVIOUS ITERATION
C      MEMZ(I)= THE Z-COORDINATE OF POINT I FROM THE PREVIOUS ITERATION
C      SIZE= THE STEP-SIZE
C
C      INTEGER VARIABLES:
C      N= THE NUMBER OF POINTS
C      ENDPT1= THE NUMBER ASSOCIATED WITH ENDPOINT 1
C      ENDPT2= THE NUMBER ASSOCIATED WITH ENDPOINT 2
C      NUMFIX= THE NUMBER OF POINTS THAT HAVE BEEN FIXED
C
C      CHARACTER VARIABLES:
C      STATPT(I)= THE STATUS OF POINT I; EITHER FIXED OR UNFIXED
C      STATUS= THE STATUS OF THE PROGRAM; EITHER DONE OR RUN
C
C
C*******************************************************************
C
C
      SUBROUTINE FIXPT(X,Y,Z,N,STATPT,ENDPT1,ENDPT2,SIZE,STATUS,MEMX,
     &                 MEMY,MEMZ)
      REAL  X(50),Y(50),Z(50),DUM1,DUM2(2),DUM3(2),DUM4(2),MEMX(50),
     &      MEMY(50),MEMZ(50),SIZE
      INTEGER  N,ENDPT1,ENDPT2,NUMFIX
      CHARACTER  STATPT(50)*1,STATUS*4
C
C                       -- IMPORTANT NOTE TO USER --
C
C      IF THE USER WISHES TO CHANGE THE MAXIMUM STEP-SIZE, HE MUST MAKE
```

```
C      THE CHANGE HERE AS WELL AS IN THE MAIN PROGRAM.
C
       SIZE=130.0
       NUMFIX=0
C  THIS LOOP COUNTS THE NUMBER OF POINTS THAT ARE ALREADY FIXED
       DO 600 I=1,N
            IF (STATPT(I) .EQ. 'F') THEN
                 NUMFIX=NUMFIX+1
            END IF
600    CONTINUE
       CALL FINDD(X,Y,Z,N,STATPT,DUM1,DUM2,DUM3,DUM4,ENDPT1,ENDPT2,
      & STATUS)
C  THESE NEXT TWO IF STATEMENTS FIX A NEW POINT
       IF (STATPT(ENDPT1) .EQ. 'U') THEN
            NUMFIX=NUMFIX+1
            STATPT(ENDPT1)='F'
            GO TO 610
       END IF
       IF (STATPT(ENDPT2) .EQ. 'U') THEN
            NUMFIX=NUMFIX+1
            STATPT(ENDPT2)='F'
       END IF
C  IF ALL THE POINTS ARE FIXED, THEN CHECK TO SEE IF THEY ARE IN THE
C  SAME LOCATION AS BEFORE
610    IF (NUMFIX .EQ. N) THEN
            DO 620 I=1,N
                 IF (MEMX(I) .NE. X(I) .OR. MEMY(I) .NE. Y(I) .OR.
      &           MEMZ(I) .NE. Z(I)) THEN
                      STATUS='RUN'
                      GO TO 630
                 ELSE
                      STATUS='DONE'
                 END IF
620         CONTINUE
C  IF THE POINTS HAVE MOVED SINCE THE LAST ITERATION, THEN UNFIX ALL
C  THE POINTS, STORE THEIR LOCATIONS IN MEMORY, AND RUN THROUGH THE
C  PROGRAM AGAIN
630         DO 640 I=1,N
                 STATPT(I)='U'
                 MEMX(I)=X(I)
                 MEMY(I)=Y(I)
                 MEMZ(I)=Z(I)
640         CONTINUE
       END IF
       END
C
C
C*****************************************************************
C
C                         SUBROUTINE PRINT
C
C           THIS SUBROUTINE PRINTS THE LOCATION OF ALL N POINTS TO THE
```

```
C       FILE CALLED 'SPCOUT' (I LIKE TO CALL IT 'SPACE OUT').
C
C
C*****************************************************************************
C
C                               VARIABLE DEFINITIONS
C
C       REAL VARIABLES:
C       X(I)= THE X-COORDINATE OF POINT I
C       Y(I)= THE Y-COORDINATE OF POINT I
C       Z(I)= THE Z-COORDINATE OF POINT I
C
C       INTEGER VARIABLES:
C       N= THE NUMBER OF POINTS
C
C
C*****************************************************************************
C
C
        SUBROUTINE PRINT(X,Y,Z,N)
        REAL  X(50),Y(50),Z(50)
        INTEGER  N
        WRITE(3,700)
        DO 710 I=1,N
             WRITE(3,720)I,X(I),Y(I),Z(I)
710     CONTINUE
        WRITE(3,730)
700     FORMAT('  POINT',9X,'L*',13X,'U*',13X,'V*')
720     FORMAT(3X,I2,7X,F8.3,7X,F8.3,7X,F8.3)
730     FORMAT(' ')
        END
C
C
C*****************************************************************************
C
C                               SUBROUTINE RGBOUT
C
C            THIS SUBROUTINE CONVERTS POINTS FROM THEIR L*, U*, AND V*
C       COORDINATES TO THEIR TRISTIMULUS VALUES AND THEN TO THEIR
C       PHOSPHOR LUMINANCES AND PRINTS THESE VALUES TO THE FILE NAMED
C       'SPCOUT' (A.K.A. SPACE-OUT).  THE USER MAY CHOOSE TO ESTABLISH
C       DIFFERENT VALUES OF MAXIMUM LUMINANCE, U-PRIME NOT, V-PRIME NOT,
C       AND CHROMATICITY COORDINATES FOR THE GUNS DEPENDING ON HIS OR
C       HER PARTICULAR APPLICATION.  ALSO, FOR SPACING IN A SYSTEM
C       OTHER THAN THE CIE L*U*V* SYSTEM, DIFFERENT TRANSFORMATION
C       EQUATIONS MUST BE USED.
C
C
C*****************************************************************************
C
C                            VARIABLE DEFINITIONS
C
```

```
C       REAL VARIABLES:
C       LSTAR(I)= THE VALUE OF L* FOR POINT I
C       USTAR(I)= THE VALUE OF U* FOR POINT I
C       VSTAR(I)= THE VALUE OF V* FOR POINT I
C       XR= THE X CHROMATICITY COORDINATE FOR THE RED GUN
C       YR= THE Y CHROMATICITY COORDINATE FOR THE RED GUN
C       XG= THE X CHROMATICITY COORDINATE FOR THE GREEN GUN
C       YG= THE Y CHROMATICITY COORDINATE FOR THE GREEN GUN
C       XB= THE X CHROMATICITY COORDINATE FOR THE BLUE GUN
C       YB= THE Y CHROMATICITY COORDINATE FOR THE BLUE GUN
C       YNOT= THE MAXIMUM LUMINANCE ALLOWABLE
C       VPMNOT= THE VALUE OF V-PRIME NOT
C       UPMNOT= THE VALUE OF U-PRIME NOT
C       XTOTAL(I)= THE X-COORDINATE OF THE TRISTIMULUS VALUE FOR POINT I
C       YTOTAL(I)= THE Y-COORDINATE OF THE TRISTIMULUS VALUE FOR POINT I
C       ZTOTAL(I)= THE Z-COORDINATE OF THE TRISTIMULUS VALUE FOR POINT I
C       UPRIME(I)= THE VALUE OF U-PRIME FOR POINT I
C       VPRIME(I)= THE VALUE OF V-PRIME FOR POINT I
C       SMALLX(I)= THE X CHROMATICITY COORDINATE OF POINT I
C       SMALLY(I)= THE Y CHROMATICITY COORDINATE OF POINT I
C       SMALLZ(I)= THE Z CHROMATICITY COORDINATE OF POINT I
C       YRED(I)= THE RED PHOSPHOR LUMINANCE FOR POINT I
C       YGREEN(I)= THE GREEN PHOSPHOR LUMINANCE FOR POINT I
C       YBLUE(I)= THE BLUE PHOSPHOR LUMINANCE FOR POINT I
C       K1= THE FIRST-ROW, FIRST-COLUMN VALUE OF THE TRANSFORMATION MATRIX
C           THAT CONVERTS TRISTIMULUS VALUES TO PHOSPHOR LUMINANCES
C       K2= THE FIRST-ROW, SECOND-COLUMN VALUE OF THE MATRIX
C       K3= THE FIRST-ROW, THIRD-COLUMN VALUE OF THE MATRIX
C       K4= THE THIRD-ROW, FIRST-COLUMN VALUE OF THE MATRIX
C       K5= THE THIRD-ROW, SECOND-COLUMN VALUE OF THE MATRIX
C       K6= THE THIRD-ROW, THIRD-COLUMN VALUE OF THE MATRIX
C       DET= THE DETERMINANT OF THE TRANSFORMATION MATRIX
C
C       INTEGER VARIABLES:
C       N= THE NUMBER OF POINTS
C
C       CHARACTER VARIABLES:
C       STATUS= THE STATUS OF THE PROGRAM; EITHER 'DONE' OR 'RUN'
C
C
C*****************************************************************
C
C
        SUBROUTINE RGBOUT(LSTAR,USTAR,VSTAR,N,YRED,YGREEN,YBLUE,STATUS)
        REAL  LSTAR(50),USTAR(50),VSTAR(50),XR,YR,XG,YG,XB,YB,YNOT,
      %       VPMNOT,UPMNOT,XTOTAL(50),YTOTAL(50),ZTOTAL(50),UPRIME(50),
      %       VPRIME(50),SMALLX(50),SMALLY(50),SMALLZ(50),YRED(50),
      %       YGREEN(50),YBLUE(50),K1,K2,K3,K4,K5,K6,DET
        INTEGER  N
        CHARACTER  STATUS*4
        XR=.6
        YR=.3
      XG=.3
      YG=.58
      XB=.13
      YB=.08
      YNOT=230.0
      UPMNOT=.195376
      VPMNOT=.469700
      K1=XR/YR
      K2=XG/YG
      K3=XB/YB
      K4=(1-XR-YR)/YR
      K5=(1-XG-YG)/YG
      K6=(1-XB-YB)/YB
      DET=K1*K6-K1*K5+K2*K4-K2*K6+K3*K5-K3*K4
      DO 800 I=1,N
C   THESE ARE THE TRANSFORMATION EQUATIONS
          YTOTAL(I)=(((LSTAR(I)+16)/116)**3)*YNOT
          UPRIME(I)=(USTAR(I)/(13*LSTAR(I)))+UPMNOT
          VPRIME(I)=(VSTAR(I)/(13*LSTAR(I)))+VPMNOT
          SMALLX(I)=9*UPRIME(I)/(6*UPRIME(I)-16*VPRIME(I)+12)
          SMALLY(I)=4*VPRIME(I)/(6*UPRIME(I)-16*VPRIME(I)+12)
          SMALLZ(I)=1-SMALLY(I)-SMALLX(I)
          XTOTAL(I)=SMALLX(I)*YTOTAL(I)/SMALLY(I)
          ZTOTAL(I)=SMALLZ(I)*YTOTAL(I)/SMALLY(I)
          YRED(I)=(XTOTAL(I)*(K6-K5)+YTOTAL(I)*(K3*K5-K2*K6)+
     &     ZTOTAL(I)*(K2-K3))/DET
          YGREEN(I)=(XTOTAL(I)*(K4-K6)+YTOTAL(I)*(K1*K6-K3*K4)+
     &     ZTOTAL(I)*(K3-K1))/DET
          YBLUE(I)=(XTOTAL(I)*(K5-K4)+YTOTAL(I)*(K2*K4-K1*K5)+
     &     ZTOTAL(I)*(K1-K2))/DET
800   CONTINUE
C   THIS SECTION FORMATS AND PRINTS THE COMPUTED TRISTIMULUS VALUES AND
C   PHOSPHOR LUMINANCES
      IF (STATUS .EQ. 'DONE') THEN
          WRITE(3,810)
          WRITE(3,820)
          DO 830 I=1,N
              WRITE(3,840)I,XTOTAL(I),YTOTAL(I),ZTOTAL(I)
830       CONTINUE
          WRITE(3,850)
          WRITE(3,860)
          WRITE(3,870)
          DO 880 I=1,N
              WRITE(3,840)I,YRED(I),YGREEN(I),YBLUE(I)
880       CONTINUE
          WRITE(3,850)
          WRITE(3,850)
      END IF
810   FORMAT(14X,'THE TRISTIMULUS VALUES ARE')
820   FORMAT(2X,'POINT',9X,'X',14X,'Y',14X,'Z')
840   FORMAT(3X,I2,7X,F8.3,7X,F8.3,7X,F8.3)
850   FORMAT(' ')
```

```
860    FORMAT(2X,'THE PHOSPHOR LUMINANCES FOR THE GUNS ARE')
870    FORMAT(2X,'POINT',8X,'RED',11X,'GREEN',11X,'BLUE')
       END
```

Appendix C. Example of Roley Heuristic Spacing Four Points in a Square

Appendix C. Example of Roley Heuristic Spacing Four Points in a Square

```
POINT     X-COORDINATE      Y-COORDINATE    Z-COORDINATE
  1             [illegible]       [illegible]       [illegible]
  2             [illegible]       [illegible]       [illegible]
  3             [illegible]       [illegible]       [illegible]
  4             [illegible]       [illegible]       [illegible]
   [illegible]
   [illegible]
   THE STEP-SIZE IS        [illegible]
POINT     X-COORDINATE      Y-COORDINATE    Z-COORDINATE
  1             [illegible]       [illegible]       [illegible]
  2             [illegible]       [illegible]       [illegible]
  3             [illegible]       [illegible]       [illegible]
  4             [illegible]       [illegible]       [illegible]
   THE MAXIMIN DISTANCE IS           [illegible]
   CORRESPONDING TO POINTS   [illegible] AND  [illegible]
   THE STEP-SIZE IS        .5[illegible]
POINT     X-COORDINATE      Y-COORDINATE    Z-COORDINATE
  1             .137              .844              .000
  2             .[illegible]28              .77[illegible]              .000
  3             .[illegible]              .[illegible]2              .000
  4             .[illegible]              .01[illegible]              .000
   THE MAXIMIN DISTANCE IS           .38[illegible]
   CORRESPONDING TO POINTS   3 AND  4
   THE STEP-SIZE IS        .5000
POINT     X-COORDINATE      Y-COORDINATE    Z-COORDINATE
  1             .[illegible]              .8[illegible]              .000
  2             .[illegible]28              .77[illegible]              .000
  3             .[illegible]              .[illegible]2              .000
  4             .[illegible]              .[illegible]              .000
   THE MAXIMIN DISTANCE IS           .49[illegible]
   CORRESPONDING TO POINTS   1 AND  2
   THE STEP-SIZE IS        .[illegible]
POINT     X-COORDINATE      Y-COORDINATE    Z-COORDINATE
  1             .[illegible]              .[illegible]              .[illegible]
  2             .1[illegible]              .[illegible]              .000
  3             .17[illegible]              .[illegible]              .000
  4             .[illegible]              .[illegible]              .000
   THE MAXIMIN DISTANCE IS           .[illegible]
   CORRESPONDING TO POINTS   [illegible]
   THE STEP-SIZE IS        .[illegible]
POINT     X-COORDINATE      Y-COORDINATE    Z-COORDINATE
  1             .[illegible]7              .[illegible]              .[illegible]
  2             .[illegible]              .[illegible]              .[illegible]
  3             .1[illegible]              .[illegible]              .000
  4             .[illegible]              .[illegible]              .000
   THE MAXIMIN DISTANCE IS           .51[illegible]
   CORRESPONDING TO POINTS   1 AND  [illegible]
   THE STEP-SIZE IS        .[illegible]
  3              .[illegible]           .[illegible]
  4              .[illegible]           .[illegible]
   THE MAXIMIN DISTANCE IS           .[illegible]
   CORRESPONDING TO POINTS  [illegible]
   THE STEP-SIZE IS        .[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-C
  1              .[illegible]           .[illegible]
  2              .[illegible]           .[illegible]
  3              .[illegible]           .[illegible]
  4              .[illegible]           .[illegible]
   THE MAXIMIN DISTANCE IS           .[illegible]
   CORRESPONDING TO POINTS  [illegible]
   THE STEP-SIZE IS        .1[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COO
  1              .887           .7[illegible]
  2              .[illegible]           .[illegible]1
  3              .01[illegible]           .077
  4              .80[illegible]           .01[illegible]
   THE MAXIMIN DISTANCE IS           .58[illegible]
   CORRESPONDING TO POINTS  1 AND  4
   THE STEP-SIZE IS        .12[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COOR
  1              .887           .71[illegible]
  2              .253           .9[illegible]1
  3              .010           .077
  4              .675           .015
   THE MAXIMIN DISTANCE IS           .65[illegible]
   CORRESPONDING TO POINTS  1 AND  [illegible]
   THE STEP-SIZE IS        .12[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-CO
  1              .887           .71[illegible]
  2              .128           .9[illegible]1
  3              .010           .077
  4              .575           .015
   THE MAXIMIN DISTANCE IS           .[illegible]
   CORRESPONDING TO POINTS  [illegible]
   THE STEP-SIZE IS        .1[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-CO
  1              .887           .71[illegible]
  2              .128           .9[illegible]1
  3              .010           .077
  4              .[illegible]           .01[illegible]
   THE MAXIMIN DISTANCE IS           .7[illegible]
   CORRESPONDING TO POINTS  1 AND  [illegible]
   THE STEP-SIZE IS        .1[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-CO
  1              .887           .844
  2              .128           .901
  3              .010           .077
  4              .80[illegible]           .01[illegible]
```

```
   THE MAXIMIN DISTANCE IS        [illegible]
   CORRESPONDING TO POINTS  [illegible] AND  [illegible]
   THE STEP-SIZE IS       .[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1           .[illegible]          .[illegible]          .[illegible]
  2           .[illegible]          .[illegible]          .[illegible]
  3           .[illegible]          .[illegible]          .[illegible]
  4           .[illegible]          .[illegible]          .[illegible]
   THE MAXIMIN DISTANCE IS        .[illegible]
   CORRESPONDING TO POINTS  [illegible] AND  [illegible]
   THE STEP-SIZE IS       .[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1           .[illegible]7          .[illegible]          .[illegible]
  2           .[illegible]          .[illegible]          .[illegible]
  3           .010          .977          .[illegible]
  4           .[illegible]          .140          .[illegible]
   THE MAXIMIN DISTANCE IS        .[illegible]
   CORRESPONDING TO POINTS  2 AND  3
   THE STEP-SIZE IS       .1250
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1           .887          .959          .000
  2           .003          .991          .000
  3           .010          .977          .000
  4           .925          .140          .000
   THE MAXIMIN DISTANCE IS        .824
   CORRESPONDING TO POINTS  2 AND  3
   THE STEP-SIZE IS       .0625
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1           .887          .969          .000
  2           .003          .953          .000
  3           .972          .014          .000
  4           .925          .140          .000
   THE MAXIMIN DISTANCE IS        .850
   CORRESPONDING TO POINTS  1 AND  4
   THE STEP-SIZE IS       .0625
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1           .[illegible]          .[illegible]          .000
  2           .[illegible]          .[illegible]          .[illegible]
  3           .972          .014          .000
  4           .987          .977          .000
   THE MAXIMIN DISTANCE IS        .[illegible]
   CORRESPONDING TO POINTS  1 AND  4
   THE STEP-SIZE IS       .[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1           .[illegible]          .[illegible]          .[illegible]
  2           .[illegible]          .[illegible]          .[illegible]
  3           .972          .014          .000
  4           .987          .015          .000
   THE MAXIMIN DISTANCE IS        .91[illegible]
   CORRESPONDING TO POINTS  3 AND  4
   THE STEP-SIZE IS       .0625
```

```
                     [illegible]             [illegible]         .
  4                  [illegible]             [illegible]         .
   THE MAXIMIN DISTANCE IS [illegible]
   CORRESPONDING TO POINTS  1 [illegible]
   THE STEP-SIZE IS [illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1          [illegible]      [illegible]      [illegible]
  2          [illegible]      [illegible]      [illegible]
  3          [illegible]      [illegible]      [illegible]
  4          [illegible]      [illegible]      [illegible]
   THE MAXIMIN DISTANCE IS        [illegible]
   CORRESPONDING TO POINTS  1 [illegible]
   THE STEP-SIZE IS      [illegible]
POINT   X-CO[illegible]      Y-CO[illegible]      Z-CO[illegible]ATE
  1          .[illegible]8[illegible]        .[illegible]        .[illegible]
  2          [illegible]         .[illegible]        .[illegible]
  3          .[illegible]1[illegible]        .[illegible]14       .[illegible]
  4          .[illegible]7        .[illegible]1[illegible]       .[illegible]
   THE MAXIMIN DISTANCE IS        .9[illegible]
   CORRESPONDING TO POINTS  1 AND  4
   THE STEP-SIZE IS      .0313
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1          .98[illegible]        .[illegible]        .[illegible]
  2          .003         .9[illegible]5        .00[illegible]
  3          .010         .[illegible]14        .00[illegible]
  4          .987         .[illegible]1[illegible]        .0[illegible]
   THE MAXIMIN DISTANCE IS        .9[illegible]
   CORRESPONDING TO POINTS  1 A[illegible]
   THE STEP-SIZE IS      .015[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1          .9[illegible]        .98[illegible]        .[illegible]
  2          .[illegible]         .9[illegible]         .00[illegible]
  3          .01[illegible]        .[illegible]14        .0[illegible]
  4          .987         .[illegible]        .0[illegible]
   THE MAXIMIN DISTANCE IS        .9[illegible]
   CORRESPONDING TO POINTS  1 A[illegible]
   THE STEP-SIZE IS      .[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1          .[illegible]         .[illegible]         .[illegible]
  2          .[illegible]         .[illegible]         .[illegible]
  3          .[illegible]         .[illegible]         .[illegible]
  4          .[illegible]         .[illegible]         .
   THE MAXIMIN DISTANCE IS        .[illegible]
   CORRESPONDING TO POINTS  1 [illegible]
   THE STEP-SIZE IS      .[illegible]
POINT   X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1          .[illegible]         .[illegible]         .[illegible]
  2          .[illegible]         .9[illegible]        .[illegible]
  3          .[illegible]1[illegible]        .[illegible]1[illegible]        .[illegible]
  4          .[illegible]         .[illegible]         .[illegible]
  [illegible]
  CORRESPONDING TO POINTS [illegible]
  THE STEP-SIZE IS       .[illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1          .[illegible]          .[illegible]          .[illegible]
  2          .[illegible]          .[illegible]          .[illegible]
  3          .[illegible]          .[illegible]          .[illegible]
  4          .[illegible]          .[illegible]          .[illegible]
  THE MAXIMIN DISTANCE IS       .[illegible]
  CORRESPONDING TO POINTS  1 [illegible]
  THE STEP-SIZE IS       .[illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1          .[illegible]          .[illegible]          .[illegible]
  2          .[illegible]03          .[illegible]          .[illegible]
  3          .[illegible]1[illegible]          .[illegible]7          .[illegible]
  4          .[illegible]          .0[illegible]7          .[illegible]
  THE MAXIMIN DISTANCE IS       .[illegible]
  CORRESPONDING TO POINTS  3 AND  4
  THE STEP-SIZE IS       .0078
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1          .996          .993          .[illegible]
  2          .003          .995          .00[illegible]
  3          .002          .007          .00[illegible]
  4          .995          .007          .0[illegible]
  THE MAXIMIN DISTANCE IS       .98[illegible]
  CORRESPONDING TO POINTS  1 AND  4
  THE STEP-SIZE IS       .0078
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1          .996          .993          .[illegible]
  2          .003          .995          .00[illegible]
  3          .00[illegible]          .007          .0[illegible]
  4          .995          .007          .00[illegible]
  THE MAXIMIN DISTANCE IS       .986
  CORRESPONDING TO POINTS  1 AND  4
  THE STEP-SIZE IS       .0039
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1          .[illegible]          .[illegible]          .[illegible]
  2          .[illegible]          .99[illegible]          .[illegible]
  3          .002          .[illegible]7          .0[illegible]
  4          .999          .[illegible]3          .00[illegible]
  THE MAXIMIN DISTANCE IS       .98[illegible]
  CORRESPONDING TO POINTS  2 AND  3
  THE STEP-SIZE IS       .[illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1          .[illegible]          .[illegible]          .[illegible]
  2          .[illegible]          .9[illegible]          .[illegible]
  3          .[illegible]          .[illegible]3          .00[illegible]
  4          .99[illegible]          .003          .00[illegible]
  THE MAXIMIN DISTANCE IS       .989
  CORRESPONDING TO POINTS  1 AND  2
  THE STEP-SIZE IS       .0039
[illegible]
```

```
  3           [illegible]           [illegible]           [illegible]
  4           [illegible]           [illegible]           [illegible]
   THE MAXIMIN [illegible] IS        [illegible]
   [illegible]
   THE STEP-SIZE [illegible]    [illegible]
POINT   X-COORDINATE     Y-COORDINATE     Z-COORDINATE
  1           [illegible]           [illegible]           [illegible]
  2           [illegible]           [illegible]           [illegible]
  3           [illegible]           [illegible]           [illegible]
  4           [illegible]           [illegible]           [illegible]
   THE MAXIMIN DISTANCE IS        [illegible]
   CORRESPONDING TO POINTS  3 AND  [illegible]
   THE STEP-SIZE IS       [illegible]
POINT   X-COORDINATE     Y-COORDINATE     Z-COORDINATE
  1           [illegible]           [illegible]           .000
  2           [illegible]           .998           [illegible]
  3           [illegible]           [illegible]           .000
  4           .997           .003           .000
   THE MAXIMIN DISTANCE IS        .993
   CORRESPONDING TO POINTS  3 AND  4
   THE STEP-SIZE IS       .0020
POINT   X-COORDINATE     Y-COORDINATE     Z-COORDINATE
  1           [illegible]           .996           .000
  2           .001           .998           .000
  3           .004           .003           .000
  4           .999           .001           .000
   THE MAXIMIN DISTANCE IS        .995
   CORRESPONDING TO POINTS  3 AND  4
   THE STEP-SIZE IS       .0020
POINT   X-COORDINATE     Y-COORDINATE     Z-COORDINATE
  1           .999           .996           .000
  2           .001           .998           .000
  3           .002           .003           .000
  4           .999           .001           .000
   THE MAXIMIN DISTANCE IS        .996
   CORRESPONDING TO POINTS  1 AND  4
   THE STEP-SIZE IS       [illegible]
POINT   X-COORDINATE     Y-COORDINATE     Z-COORDINATE
  1           [illegible]           .998           .000
  2           [illegible]           [illegible]           .000
  3           [illegible]           [illegible]           [illegible]
  4           [illegible]           [illegible]           [illegible]
   THE MAXIMIN DISTANCE IS        [illegible]
   CORRESPONDING TO POINTS  2 AND  3
   THE STEP-SIZE IS       [illegible]
POINT   X-COORDINATE     Y-COORDINATE     Z-COORDINATE
  1           [illegible]           [illegible]           .000
  2           .001           [illegible]           .000
  3           [illegible]           .001           .000
  4           [illegible]           .001           [illegible]
```

Copy available to DTIC does not permit fully legible reproduction

```
     [illegible]
     CORRESPONDING TO POINTS  1 AND [illegible]
     THE STEP-SIZE IS       .[illegible]
POINT    X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1            .[illegible]            .[illegible]            .[illegible]
  2            .[illegible]            .[illegible]            .[illegible]
  3            .[illegible]            .[illegible]            .[illegible]
  4            .[illegible]            .[illegible]            .[illegible]
     [illegible]
     [illegible]
     THE STEP-SIZE IS       .[illegible]
POINT    X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1            .[illegible]            .[illegible]            .[illegible]
  2            .[illegible]            .[illegible]            .[illegible]
  3            .[illegible]            .[illegible]1           .[illegible]
  4            .[illegible]            .[illegible]            .[illegible]
     THE MAXIMIN DISTANCE IS        .[illegible]
     CORRESPONDING TO POINTS  3 AND  4
     THE STEP-SIZE IS       .0010
POINT    X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1            .999            .998            .000
  2            .000            .999            .000
  3            .001            .002            .000
  4           1.000            .001            .000
     THE MAXIMIN DISTANCE IS        .998
     CORRESPONDING TO POINTS  1 AND  4
     THE STEP-SIZE IS       .0010
POINT    X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1            .998            .999            .000
  2            .000            .999            .000
  3            .001            .002            .000
  4           1.000            .001            .000
     THE MAXIMIN DISTANCE IS        .9[illegible]
     CORRESPONDING TO POINTS  [illegible] AND  [illegible]
     THE STEP-SIZE IS       .0010
POINT    X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1            .998            .9[illegible]            .[illegible]
  2            .[illegible]            .[illegible]            .[illegible]
  3            .002            .001            .000
  4           1.000            .001            .001
     THE MAXIMIN DISTANCE IS        .9[illegible]
     CORRESPONDING TO POINTS  1 AND [illegible]
     THE STEP-SIZE IS       .0010
POINT    X-COORDINATE    Y-COORDINATE    Z-COORDINATE
  1            .9[illegible]            .[illegible]            .[illegible]
  2            .00[illegible]            .[illegible]            .[illegible]
  3            .00[illegible]            .0[illegible]1            .[illegible]
  4           1.00[illegible]            .[illegible]1            .[illegible]
     THE MAXIMIN DISTANCE IS        .9[illegible]
     CORRESPONDING TO POINTS  3 AND  4
     THE STEP-SIZE IS       .0010
```

```
[illegible]
 THE MAXIMIN DISTANCE IS        [illegible]
 CORRESPONDING TO POINTS  [illegible]
 THE STEP-SIZE IS      [illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
[illegible]
 THE MAXIMIN DISTANCE IS        [illegible]
 CORRESPONDING TO POINTS  [illegible]
 THE STEP-SIZE IS      [illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1            [illegible]     1.[illegible]          .[illegible]
  2            [illegible]      .[illegible]          .[illegible]
  3            .[illegible]1      .[illegible]1          .[illegible]
  4           1.000          .000          .000
 THE MAXIMIN DISTANCE IS        .99[illegible]
 CORRESPONDING TO POINTS  2 AND [illegible]
 THE STEP-SIZE IS      .0005
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1            .999         1.000          .000
  2            .000          .999          .000
  3            .001          .000          .000
  4           1.000          .000          .000
 THE MAXIMIN DISTANCE IS        .99[illegible]
 CORRESPONDING TO POINTS  1 AND 2
 THE STEP-SIZE IS      .00[illegible]8
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1           1.00[illegible]         1.00[illegible]          .[illegible]
  2            .00[illegible]          .[illegible]          .[illegible]
  3            .[illegible]1          .[illegible]          .[illegible]
  4           1.[illegible]          .[illegible]          .00[illegible]
 THE MAXIMIN DISTANCE IS        .[illegible]
 CORRESPONDING TO POINTS  [illegible]
 THE STEP-SIZE IS      .[illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1           1.[illegible]         1.[illegible]          .[illegible]
  2            .[illegible]          .[illegible]          .[illegible]
  3            .[illegible]          .[illegible]          .[illegible]
  4           1.[illegible]          .[illegible]          .[illegible]
 THE MAXIMIN DISTANCE IS        .[illegible]
 CORRESPONDING TO POINTS  [illegible]
 THE STEP-SIZE IS      .[illegible]
POINT   X-COORDINATE   Y-COORDINATE   Z-COORDINATE
  1           1.[illegible]         1.[illegible]          .[illegible]
  2            .[illegible]         1.0[illegible]          .[illegible]
  3            .[illegible]          .[illegible]          .[illegible]
  4           1.[illegible]          .[illegible]          .[illegible]
```

```
[illegible]
```

Although the solution is now optimal, the program continues until the step-size is less than .0001 at which time the step-size is reset back to .5 and point number three is fixed in its place, allowing the algorithm to concentrate on trying to optimize the other three points. The step-size then gradually decreases back to .0001 and another point is fixed, and so on, until all four points are fixed. At that time, the locations of the points are stored in memory and all the points are unfixed to see if any more improvements can be made. The program then goes through the whole process again, reducing then resetting the step-size and fixing the points until all four points have been fixed. Because no improvements could be made, the program is done at that time.

Bibliography

1. Calamai, Paul and Christakis Charalambous. "Solving Multifacility Location Problems Involving Euclidean Distances," Naval Research Logistics Quarterly, 27: 609-620 (December 1980).

2. Carter, Robert C. and Ellen G. Carter. "High-Contrast Sets of Colors," Applied Optics, 21: 2936-2939 (August 1982).

3. Charalambous, Christakis. "An Iterative Method for the Multifacility Location Allocation Problem with Euclidean Distances," Naval Research Logistics Quarterly, 28: 325-337 (June 1981).

4. Chen, Reuven. "Solution of Minisum and Minimax Location-Allocation Problems with Euclidean Distances," Naval Research Logistics Quarterly, 30: 449-459 (March 1983).

5. Church, Richard L. and Robert S. Garfinkel. "Locating an Obnoxious Facility on a Network," Transportation Science, 12: 107-118 (May 1978).

6. Cooper, Leon. "Location-Allocation Problems," Operations Research, 11: 331-343 (1963).

7. Griffith, R. E. and R. A. Stewart. "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," Management Science, 7: 379-392 (1961).

8. Juel, Henrik. "A Note on Solving Multifacility Location Problems Involving Euclidean Distances," Naval Research Logistics Quarterly, 29: 179-180 (March 1982).

9. Kirtpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi. "Optimization by Simulated Annealing," Science, 220: 671-679 (May 1983).

10. Love, Robert F. "One-Dimensional Facility Location-Allocation Using Dynamic Programming," Management Science, 22: 614-617 (January 1976).

11. Love, Robert F. and Henrik Juel. "Properties and Solution Methods for Large Location-Allocation Problems," Journal of the Operational Research Society of America, 33: 443-452 (May 1982).

12. Love, Robert F. and James G. Morris. "Solving Multifacility Location Problems Involving ℓ_p Distances Using Convex Programming," Operations Research, 23: 581-587 (1975).

13. Robertson, A. "The CIE 1976 Color Difference Formulae," Color Research and Application, 2: 7-11 (1977).

VITA

Captain Ross E. Roley was born on 11 September 1958 in Sterling, Illinois. In 1976, he graduated fifth in his class of approximately 300 from Sehome High School in Bellingham, Washington. That summer, he entered the United States Air Force Academy from which he received a Bachelor of Science degree in Mathematics and a commission as an Officer in the United States Air Force in May 1980, graduating in the top 15 percent of his class. Following an abortive attempt at undergraduate pilot training because of airsickness, he was assigned as a data analyst to the Air Force Tactical Communications Test Team at Fort Huachuca, Arizona, where he served from 1981 to 1983. While there, he received an Air Force Commendation Medal for his outstanding analytical support for the new generation of ground tactical communications equipment. In addition, he was instrumental in winning for the unit three straight Commander's Cup Trophies for intramural athletic excellence while serving as Unit Athletic Officer. He was subsequently selected to attend the Air Force Institute of Technology at Wright-Patterson Air Force Base, Ohio, where he entered in June of 1983 as a Master's student in Operations Research.

Permanent address: 1200 Undine St.
Bellingham, Washington 98226

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GOR/OS/84D-10	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering Air Force Institute of Technology 6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFAMRL 8b. OFFICE SYMBOL (If applicable) HEA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) AFAMRL Wright-Patterson AFB, OH 45433	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 62202F PROJECT NO.: 7184 TASK NO.: 11 WORK UNIT NO.: 49
11. TITLE (Include Security Classification) See Box 19	
12. PERSONAL AUTHOR(S) Ross E. Roley, B.S., Capt, USAF	
13a. TYPE OF REPORT MS Thesis 13b. TIME COVERED FROM ____ TO ____	14. DATE OF REPORT (Yr., Mo., Day) 1984 December 15. PAGE COUNT 90
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES

FIELD	GROUP	SUB. GR.
12	02	
20	06	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Colorimetry, Operations Research

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: A HEURISTIC SOLUTION TO THE SPACING PROBLEM
Maximizing the Minimum Euclidean Distance
Among N Points in a Convex Region

Thesis Chairman: Ivy D. Cook, Lt Col, USAF

Approved for public release: IAW AFR 190-17.
[illegible] 21 Feb 85
LYNN E. WOLAVER
Dean for [illegible] Professional Development
Air Force [illegible] (ATC)
Wright [illegible]

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Ivy D. Cook, Lt Col, USAF	22b. TELEPHONE NUMBER (Include Area Code) 513-255-2549 22c. OFFICE SYMBOL AFIT/ENS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

This report introduces a new method for solving the problem of optimally spacing points in a three-dimensional region so that their distances from each other are as great as possible. One application of the problem deals with color selection for aircraft displays where the colors are plotted as points in a three-dimensional color space and the distance between two points is directly related to the distinguishability of the two colors. The method itself is a heuristic algorithm very similar to one designed by Carter and Carter (2). The newer algorithm apparently yields similar solutions with fewer runs, but because it is more thorough, it is slower. The program was tested on problems as large as 23 points whose feasible region had seven faces. The major disadvantage of this new method is that its solutions are not guaranteed to be optimal. As a result, the user must perform several replications of various randomly selected starting locations in order to increase the chances of achieving an optimal solution.